

El yazması geçerliliğinde

KAHRAMANOV Şirzad Şamhal oğlu

OTOMATLAŞTIRILMIŞ KONTROL SİSTEMLERİNDE
ERİLERİN HATASIZ OLARAK SIKIŞTIRILMASI VE AÇILMASI

dalı 05. 13.06

Otomatlaştırılmış bilgi-işlem ve kontrol
sistemleri (sanayide)

Teknik bilimler üzere

DOKTORA TEZİ

Danışman: Prof. Dr. ASKEROV T. M.

BAKU 1983

İÇİNDEKİLER

GİRİŞ.....	1
I. VERİ SIKIŞTIRMASI VE ONUN OTOMATLAŞTIRILMIŞ KONTROL SİSTEMLERİNİN ÇALIŞMASINA ETKİSİ.....	6
1.1. Otomatlaştırılmış kontrol sistemlerinde veri sıkıştırma probleminin güncelliği....	6
1.2. Problemin durumu ve çözülmesi gereken sorunlar.....	9
1.3. Biçimsel fazlalıkların araştırılması ve giderilmesi yöntemleri.....	14
1.4. Görelî fazlalıkların araştırılması ve giderilmesi yöntemleri.....	20
1.5. Serilerde elemanların tekrarlanmasından meydana gelen fazlalıkların araştırılması ve giderilmesi yöntemleri.....	25
1.6. Ddeğişik serilerde aynı elemanların bulunmasından meydana gelen fazlalıkların araştırılması ve giderilmesi yöntemleri.....	30
1.7. Aritmetik fazlalıkların araştırılması ve giderilmesi yöntemleri.....	33
1.8. Veri yapılarının analizi.....	34
1.9. Verilerin lojik ve fizik yapılarının karşılıklı dönüşüm proseslerinin kısa analizi...	40
I. Bölümün sonuçları.....	43
2. VERİ SIKIŞTIRMA YÖNTEMLERİNİN FONKSİYONEL TAM KÜMESİNİN SEÇİLMESİ.....	45
2.1. Veri sıkıştırma yöntemlerinin karşılaştırmalı analizi.....	45
2.2 Veri sıkıştırma yöntemlerinin fonksiyonel tam kümelerinin belirlenmesi.....	48
2.3. Veri sıkıştırma yöntemlerinin fonksiyonel tam kümelerinin sıkma başarısı açısından karşılaştırılması	52
2.4. Kortejleri iki taraflı kesilme yöntemi	62
2. Bölümün sonuçları.....	71

3. VERİ SIKIŞTIRMA PROSESLERİNİN ALGORİTMALAŞTIRILMASI, BU ALGORİTMALARIN BELİRLİ PROSEDÜRLERE AYRILMASI VE SIKMA VE AÇMA İÇİN BİRLEŞMİŞ ALGORİTMALARIN TERTİPLANMESİ.....	72
3.1. Ayırıştırıcı işaretler (AİY) ve sınıflandırıcı diyagramlar (SDY) yöntemlerinin algoritmaları.....	72
3.2. Artışlar yönteminin (AY) algoritması.....	77
3.3. AİY-C, SDY-C FY1-C ve FY2-C yöntemleri ile kortejleri kesme algoritmalarının birleştirilmesi.....	79
3.4. SKY1-C ve SKY2-C yöntemlerinin algoritmalarının birleştirilmesi.....	85
3.5. SKY1-R ve SKY2-R açma yöntemlerinin algoritmaları.....	89
3.6. Birleştirilmiş algoritmalar için başarı değerlendirmeleri.....	94
3.7. SKY ile veri sıkıştırma oranının değerlendirilmesi.....	96
3. Bölümün sonuçları.....	105
4. GELİŞTİRİLMİŞ OLAN VERİ SIKIŞTIRMA VE HATASIZ OLARAK AÇMA YÖNTEMLERİNİN SANAYİ MÜESSİSELERİNİN PASAPORTLARINI OTOMATİK OLARAK DÜZENLEYEN VE İŞLEYEN SİSTEMDE UYGULANMASI.....	106
4.1. Sanayi müessiselerinin pasaportlarını otomatik olarak düzenleyen ve işleyen sistemde veri tabanının fizik organizasyonu.....	106
4.2. Hedef veri tabanında veri sıkıştırması.....	108
4.3. Verileri sıkıştırma ve açma programları.....	112
4.4. Sıkıştırma yöntemlerinin uygulanmasından elde edilen ekonomik tasarruf.....	115
4. Bölümün sonuçları.....	121
KAYNAKLAR.....	122
EK.....	

G İ R İ Ő

Sovyetler Birlięinin 1981-1985 yılar zaman aralıęında ve 1990 yıla kadar iktisadi ve sosyal gelişme istikametleri, ekonominin bütün alanlarında denetim ve yönetimin seviyesini yükseltmek, gayretleri en iyi sonuçlara yönelmek ve bu gibi önemli problemleri ön plana çıkarmıştır. İlgili belgenin "Bilimin geliştirilmesi ve teknik seviyenin yükseltilmesi" bölümünde bilgisayar bilim ve teknolojisinin, onun maddi-malzeme temelini, matematik ve program desteęinin, ve bilgi toplama, alma ve gönderme araç ve yöntemlerinin geliştirilmesi ülkenin geleceęi için en önemli problemlerinden biri olarak gösterilmiştir[1].

Yönetimin geliştirilmesi yöntemlerinden biri müesseseler, alanlar, bakanlıklar, bölgeler vs. çerçevesinde geliştirilmiş olan otomatlaştırılmış idare sistemlerinin uygulanmasıdır. Bu yaklaşım, gittikçe daha geniş alan kapsama yöntemi ile devlet otomatlaştırılmış yönetim sistemine kadar bir geliştirme mantıęına dayanmaktadır. Böle büyük çaplı otamalaştırılmış yönetim sistemleri çok büyük miktarda veri tutma ve işleme gerektirmektedir. [2,3]'te verilen bilgilere göre bu miktar 10^7 - 10^{12} karaktere kadar varabilmektedir.

Yakın zamanda (yıl 1981) yapılmış kesin hesaplamalara göre, ekonominin ülke (Sovyetler Birlięi) çapında yönetilmesi için gereken otomatlaştırılmış kontrol sisteminin hesaplama gücü 10^{13} işlem/s, bu gücü sağlayacak olan sistemdeki makinelerin (IBM360/370 veya eşdeęerleri) sayısı 10^4 , her bir makinenin performansı 10^8 - 10^9 işlem/s, esas belleęinin kapasitesi 10^8 - 10^9 bayt, arşiv belleęinin kapasitesi 10^{10} - 10^{12} bayt, iletişim hızı ise 10^6 - 10^7 bayt/s'den az olmamalıdır[2].

Otomatlaştırılmış kontrol sistemlerinin (OKS) performansının yükseltilmesi yollarından biri, bilgisayar kaynaklarının ve iletişim araçların imkanlarının genişletilmesi,

bunun özel yollarından biri ise tutulmakta olan ve iletilen verilerin sıkışıklığının mümkün olduğu kadar yükseldilmesidir [4,5]. Bu sıkışıklık işe verilerde bulunan, fakat bellekte tutma ve işleme açısından gereksiz olan fazlalıkların kaldırılması sayesinde elde edilebilmektedir. Söz konusu fazlalıkların kaldırılması olayına Veri Sıkıştırma denir. İlgili kaynaklarda bu kavram Verilerin Fazlalıklardan Arındırılması, Bilgi Sıkıştırma, Veri Kalıplama vs.[4,14] gibi de ifade edilmektedir. Bu çalışmada son 10-15 senede yayınlanmış olan kaynaklarda sabitleşmiş olduğu gibi Veri Sıkıştırma terimi kullanılmaktadır.

Tabii ki, uygulanmakta olan her bir veri sıkıştırma yönteminin (VSY) verimi, sıkılmakta olan veride bulunan fazlalık oranına bağlıdır. Otomatlaştırılmış kontrol sistemlerinde dolaşmakta olan verilerin analizi, bunlarda fazlalık payının çok büyük olduğunu göstermiştir [5, 11-14]. Bu yüzden, söz konusu sistemlerde veri fazlalıklarının incelenmesi problemi güncelliğini korumakta devam etmektedir [5,8,12,13].

Sistemden bilgi alan hem insan hem de makine olabileceği ve bunların mantıki imkanları değişik olabileceği için sistem onunla irtibatla olan ne varsa hepsi ile doğru iletişim yapabilecek bir donatıma sahip olmalıdır. Bu yüzden, veri sıkıştırma ve açma yöntemleri de bilgi alıcılarının bilgi işlem mantığına uygun bir şekilde düzenlenmelidir. Fakat hem insan hem de makine için değişik yorumlara yol açmayacak tek temsil şekli formal temsil olduğu için sıkıştırma ve açma yöntemleri de aldığı veriler ve verdiği sonuçlar için bilinen ciddi formal ifade araçlarını kullanmalıdırlar[16,19]. Fakat, klasik enformasyon teorisi bu bakımdan yetersiz kalmaktadır. Çünkü bu teoriye göre bilgi alma, gerçek alıcının ferdi imkanlarına değil, ideal alıcının soyut potansiyel imkanlarına dayanmaktadır [19]. Teorinin ideal, pratiğin ise gerçek bilgi alıcı potansiyeline dayanması durumları arasında kalan boşluğu doldurmak için çok sayıda veri sıkıştırma yöntemi teklif edilmiştir [4-14, 20-22]. Fakat galiba ki, son zamanlara kadar büyük veri tabanlarının

olmaması, bu yöntemlere gereken dikkatin verilmemesine getirip çıkarmıştır. Bu yüzden, var olan yöntemlerden her biri yalnız belli bir ortamda sayısal olarak değerlendirilmiş, teorik olarak hangi yöntemin hangi bir ortamda nasıl davranacağı ise araştırılmamıştır. Fakat Marks'ın dediği gibi, değişik şeyler, yalnız hepsi için aynı olan bir nitelik açısından sayısal olarak değerlendirilebilirler. Yani, veri sıkıştırma yöntemleri için ağırlıklı nitelikler belirlenmeli ve yalnız bundan sonra değişik yöntemler bu niteliklerinin sayısal değerlerine göre mukayese edilmelidirler. Söz konusu nitelikler olarak ilgili kaynaklarda aşağıdakiler teklif edilmektedirler: dönüştürülmeye tabi tutulan birimlerin seviyesi [5,8], dönüştürme prosedürünün karakteri [5,8,12, 22], yöntemin evrenselliğini [5,8], giderilen fazlalığın türü [14]. Fakat bizce gösterilen niteliklerden hiç de az önemli olmayan başka nitelikler de vardır ki, bunların dikkate alınması gerek teorik gerekse de pratik açıdan çok faydalı olabilir. Bir yöntemin kullandığı giriş verisinin ve ürettiği çıkış verisinin yapısı(strüktürü) ve ürettiği fazlalık türü şu ana kadar dikkatten kaçan niteliklere en önemli örnekler olabilirler. Bu nitelikler, özellikle değişik yöntemler arasında olması gereken veri alışverişinin doğru, ciddi ve formal bir şekilde organize edilebilmesi açısından çok faydalı olabilirler.

Bu tez çalışmasının amacı, verilerde bulunan yapısal (sructural) fazlalıkların araştırılması, bu fazlalıkları gideren yöntemlerin sistematik bir şekilde incelenmesi ve yöntemler kümesinin niteliklere göre eşdeğer sınıflara bölünmesi yolu ile aşağıdaki pratik problemlerin çözülmesidir:

- Mevcut yöntemlerin yetersiz olduğu fazlalıklar için yeni yöntemlerin geliştirilmesi
- Sıkıştırma yöntemlerinin çok sayıda fazlalık tipleri bulunduran verilere uygulanma sırasının belirlenmesi
- Ortak prosedürler bulunduran algoritmaları birleştirme yolu ile mümkün olduğu kadar kompakt, fakat esnek ve etkin bir veri sıkma/açma paketinin elde edilmesi

Bu problemlerin çözülmesi sonucunda minimum sayıda ve ciddi bir sıra ile uygulanması gereken yöntemler içeren fonksiyonel tam paketler elde edilmiştir.

Tez çalışmasında araştırma araçları olarak aşağıdakilerden yararlanılmıştır:

- Fazlalık tiplerinin analizi için - veri tabanlarının ve veri yapılarının analiz ve sentezi, enformasyon teorisi, set teorisi ve matematik mantık
- Sıkıştırma yöntemlerinin eşdeğer sınıflarının ve fonksiyonel tam kümelerinin belirlenmesi için - anahtar fonksiyonlar teorisi, uygulamalı matematik, olasılık teorisi
- Sıkıştırma yöntemlerinin uygulanma sırası için - küme teorisi, graf teorisi
- Sıkıştırma yöntemlerinin algoritmalarının birleştirilmesi için - algoritma teorisi, matematik mantık, programlama teorisi, mikroprogramlı otomatların sentezi teorisi
- Sıkıştırma yöntemlerinin başarı katsayısını belirlemek için - kombinatorik, küme teorisi, ihtimal teorisi

Tez çalışması 4 esas ve 1 ek bölümden oluşmuştur (435...sayfa metin, 5 resim ve 28 tablo).

Birinci bölümde verilerde bulunan değişik fazlalık tiplerinin meydana gelme sebepleri incelenmiş, OKS'de veri sıkıştırmanın gereği açıklanmış, veri sıkıştırma problemini durumu incelenmiş ve bu problemin çözüm yolları ele alınmıştır. Bu bölümde değişik fazlalık sınıfları ve tipleri gözden geçirilmiş, değişik sıkıştırma yöntemleri incelenmiş ve fazlalık tipleri ile sıkıştırma yöntemleri arasında var olan ilişkiler tespit edilmiştir. Her bir sıkıştırma yöntemi için kullanılan ve üretilen veri yapıları belirlenmiş ve bu yola hangi yöntemin hangi veri yapısına uygulanabileceği ve hangi tür veri yapısı üreteceği belirlenmiştir. Ayrıca, bu tespitler, sıkıştırma/açma yöntemlerinin uygulanma sırasını da belirlemişler.

Tez çalışmasının bu bölümünde ayrıştırıcı işaretler yöntemi(AIY), sınıflandırma diyagramları yöntemi (SDY) , bit matrisleri yöntemi(BMY), serilerin uzunlukları yöntemi(SUY), ikili kodlama yöntemi(KY), *mantık-cabveli* yöntemi(MCY), adres-konum yöntemi(AKY), aritmetik yöntem($A_{t,t}$) ve farklar yöntemi ayrıntılı olarak araştırılmış ve değerlendirilmiştir.

İkinci bölümde araştırma konusu olan yöntemlerin matematik modelleri tertiplenmiş ve bu yöntemler arasında mevcut olan parametrik ilişkiler ortaya çıkarılmıştır. Bu yolla sıkıştırma yöntemlerinin fonksiyonel tam olan minimum kümelerinin bulunması ve bu yöntemlerin uygulanma sırası için bir teorik altyapı hazırlanmıştır. Bu arada yeni bir sıkıştırma yöntemine ihtiyaç olduğu gösterilmiş ve bu yöntem geliştirilmiştir.

Üçüncü bölümde veri sıkıştırma ve açma yöntemlerinin algoritmaları tertiplenmiş ve bu algoritmalar tipik modüllere ayrılmıştır. Gerçekleştirme kolaylığı amacı ile, bu algoritmalar günün en gelişmiş ve yaygın makineleri olan EC-EBM (IBM 360/370'in mimari eşdeğeri) sisteminin assembler diline yönelik olarak tertiplenmiştir. Bu bölümün sonunda, alınmış olan sonuçların değerlendirilmesi verilmiştir.

Dördüncü bölümde , geliştirilmiş olan minimal sıkıştırma kümesini pratik olarak uygulama problemleri ele alınmıştır. Bu yöntemler Azerbaycan Petrol-Kimya bakanlığına, "Soyuzneftemash" birliğine ve "Azerneft" birliğine bağlı olan 30 müesseseyi kapsamakta olan bir OKS'in veri tabanına uygulanmış ve donanım sarfiyatı ve makine zamanı tasarrufu açısından bir yılda yaklaşık olarak \$60000 ekonomik tasarruf sağlamıştır.

Sovyetler Birliği bakanlar kuruluna bağlı olan alanlar arası komisyon bu sistemin ekonomik problemlere yönelik OKS'ler için örnek olarak kabul edilmesi hakkında karar vermiştir.

Tez çalışması sırasında elde edilmiş olan esas sonuçlar 4 makale [23-26], 4 bildiri [26-29] ve 2 rapor [30-31] olarak yayınlanmıştır.

I.. VERİ SIKIŞTIRMASI VE ONUN OTOMATLAŞTIRILMIŞ KONTROL SİSTEMLERİNİN ÇALIŞMASINA ETKİSİ

1.1. Otomatlaştırılmış kontrol sistemlerinde veri sıkıştırma probleminin güncelliği

Bilindiği gibi, ekonominin geliştirilmesi için tercih edilen esas yollardan biri, bölgesel üretim sistemleridir. Burada amaç, bölgesel ekonomileri bir sistem olarak yönetmekle bölgelerarası dengeyi sağlamakla ekonominin bir bütün halinde denetim ve yönetim altına alınmasıdır [32]. Kontrolün bölgeler üzere gerçekleştirilmesi ihtiyacı, otomasyona tabi tutulması gereken objelerin gittikçe çoğalması ve büyümesi ve bu yüzden onların tek merkezden kontrol edilmesinin her geçen gün biraz daha zorlaşmasından doğmuştur [32]. İleride, bölgesel OKS'lerin ülke üzere tek bir dağıtılmış bilgisayar sisteminin esas bileşenleri olacağı düşünülmektedir. Bu sistem, tahmini olarak aşağıdaki parametrelere sahip olacaktır [33]:

- Kişisel terminal sayısı : 20-100 milyon
- Müesseselerdeki terminal sayısı: 100-200 milyon
- Bilgi tutma ve işleme noktalarının sayısı: 20-30 bin
- Toplam hesaplama gücü: $5 \times (10^{12} - 10^{13})$ işlem/saniye
- Bir işlemin parasal değeri: \$ 0,005-0,0005

Buradaki esas hedeflerden biri de “kağıtsız yönetim” teknolojisine ulaşmaktır. Lakin bu yolda yalnızca bilgisayarların başarısına güvenmek doğru olmaz. Çünkü hiçbir bilgisayar şartlara uygun olarak geliştirilmeden gerekeni yapamaz. Bu yüzden bilgi akış teknolojisinde, belge dolaşımında, verileri depolama yöntemlerinde, bilgisayarların mantık

ve hesaplama altyapılarında radikal gelişmeler yapmadan amaca ulaşmak imkansız gibi görünmektedir.

Halen bölgesel OKS'lerden ülke OKS'ine geçiş süreci başlanmış durumdadır. Fakat şimdiden bu sistemde çok büyük miktarda bilgi dolaşacağı, bu yüzden onların gereken hızla iletilebileceği ve işlenip hedefe sunulabileceği ile ilgili problemler çıkacağı muhakkaktır. Bu yüzden, sistem ilişkilerinin organizasyonu ön plana çıkmaktadır. Yani sistemin bilgi iletme performansı mümkün olan maksimuma , birim başına düşen fiyat ise sürekli olarak gelişmekte ve değişmekte olan teknolojik ortamda, mümkün olan minimuma ulaştırılmalıdır.

İster ülke isterse de bölgesel çaplı olsun, fark etmez, bir OKS'nin teknik, bilgisayar ve programsal araçları araziler üzere dağıtılmış olmalıdır. Bu yüzden belleklerde tutulan ve bir araziden diğer araziye iletilen veriler, manasına bir zarar gelmeden, minimuma indirilmelidir [32,34,35]. Bu amaçla kullanılabilecek en etkin araçlardan biri ise verilerin sıkıştırılmasıdır.

Bellekte tutulması veya iletilmesi gereken veri miktarının azaltılması veri sıkıştırma yöntemleri ile gerçekleştirilebilir. Fakat bir veri sıkıştırma yöntemi, yalnız fazlalık içeren verilere uygulandığında etkili olabilir. Bu yüzden OKS'lerde bulunan verilerin karakteristiklerini incelemek gerekir.

OKS'lerde kullanılan çok büyük miktarda veriler sayı, karakter ve say-karakter tip olmakla yüksek oranda değişik yapısal fazlalıklar içermektedirler. Halen esas belleği 10^8 - 10^9 bayta , arşiv belleği ise 10^{10} - 10^{12} bayta varan OKS'ler faaliyet göstermekte [2,35], bunlarda tutulan ve birimler arasında dolaşan ekonomisel veriler ise %50 ile %90 arasında fazlalık bulundurmaktadır [5]. Böyle sistemlerde veri miktarının 2-6 kat azaltılabileceği teorik ve pratik olarak gösterilmiştir [5,11,14]. Görüldüğü gibi, sıkıştırma yöntemlerini

uygulamak ve OKS'lerin veri dosyalarının hacmini önemli bir ölçüde azaltmakla iletişim yolları da dahil olmakla, sistemin performansı önemli bir ölçüde yükletilebilir [5,12,42].

Halen veri sıkıştırma yöntemleri esas belleğinin kapasitesi 10^6 - 10^7 baytı aşamayan tek bir bilgisayar çerçevesinde gerçekleştirilebilir. Fakat OKS'lerde büyüklüğü 10^8 - 10^{10} bayt olan dosyalar da var [43]. Bunun yanında, programların ve verilerin değişik sebeplerden dolayı bozulması ve kayıp olması korkusu ile yedeklenmesi gerekmektedir ki bu da problemin çok daha ağırlaşmasına sebep olmaktadır [44]. Bilgi dosyalarının bozulması olaylarının istatistik araştırılması göstermiştir ki, ayarlama, deneme ve kullanım aşamalarında bozulmuş olan veri dosyaları uygun olarak 0.7-0.95, 0.95-0.99 ve 0.99-0.9999 ihtimalle düzeltilebilir. Yalnız bunun için verilerin uygun olarak 4-5, 2-3 ve 1 kopyasının olması gerekmektedir [44]

Görüldüğü gibi, sistemde tutulmakta olan bilgilerin büyük çoğunluğu 0.0001 ihtimalle başvuru kopyalardır ve yalnız bunların sıkıştırılmış olarak tutulması problemi önemli bir ölçüde çözüme kavuşturabilir. Böylece, OKS'lerde veri sıkıştırması, aşağıdaki iki esas amaca hizmet etmektedir:

- Verileri sıkıştırmakla esas ve arşiv bellek kapasitesine olan ihtiyacı azaltmak
- Verilerin informatifliğini yükseltmekle belli bir zaman diliminde daha çok bilgi iletebilmek ve bu yolla da sistemin ortalama performansını yükseltmek

Bilindiği gibi, bölgesel OKS'lerin geliştirilmesi, bu sistemler üzerinde devlet OKS'nin kurulması ve tam gücü ile kullanılması bir 5 yıllık süreye sığmayacak kadar büyük bir iştir [2,34]. Ancak problemin zamanında çözülebilmesi için çok para ve zaman gerektiren bilgisayar şebekelerinin geliştirilmesine hemen başlamak lazımdır [34].

1.2. Problemin durumu ve çözülmesi gereken sorunlar

İnsan şuurunun temel özelliği bundan ibarettir ki, insan hayati tecrübelerini sistemleştirmekle onu kavramlar haline getirir ve her şeyi bu kavramlar üzerinden ifade etmeye çalışır. Temel olarak kavramlar 3 kategoriye bölünebilir [45]:

- *Maddi ve maddi olmayan nesneler* (objeler), mesela; müessese, çalışan, belge, proses, olay vs.
- *Nesnelerin özellikleri*, mesela; müessesenin profili, çalışanın görevi, belgenin tipi, prosesin kalitesi, olayın baş gösterme anı vs.
- *Nesneler arasındaki ilişkiler*, mesela; amir-memur , esas- yardımcı, götür-en-götürülen, önceki-sonraki, karaktersize eden – karaktersize edilen, identifikator - değer vs.

Bu kategorilerden her biri bilgisayar belleğinde identifikator adlanan bir değişken tarafından temsil edilmektedir. Bu tür identifikatorların tümünün ve onların değerlerinin bir arada bulunduğu ortam (soyut veri yapısı) *Veriler* olarak tanımlanmaktadır [5,47-49]. Veri kavramının bir başka ve daha isabetli tanımı [50]'de bulunmaktadır. Bu kaynağa göre, Veri denilen bir şey, yorumu kolay ve bir manalı olsun diye formal şekilde ifade edilmiş olan ya bir olgu, ya bir kavram, ya da bir talimat olabilir.

Nesneleri, özellikleri ve ilişkileri ifade eden değişkenlerden her biri *Bir Veri Elemanı* (VE) olarak tanımlanmaktadır. Bir bilgi sistemine ait olan bir VE o sistem çerçevesinde kullanılan en küçük birim olduğu için mantıki olarak bölünemez [51].

Belli bir nesneni karakterize eden VE'lerin tümü bir kayıt olarak, aynı tip kayıtların oluşturduğu iki boyutlu yapı file olarak tanımlanmaktadır [5,45].

Verileri matematik nesneler olarak tanımlamak için küme, kısmi düzenlenmiş küme, ciddi düzenlenmiş küme (kortej, vektör), seri, strüktür, element, birim vs. gibi kavramlardan yararlanmaktadır[5,53].

Veriler üzerinde değişik işlemler yapmak için, genellikle onlara değişik matematik nesneler olarak bakmak gerekmektedir. Mesela, bir veri tablosunu bilgisayar belleğine aktarırken, onun satır ve sütunları istenildiği gibi yer değişmeye tabi tutulabilir. Bu durumda , söz konusu tablo, kendi satırlarının veya sütunlarının düzenlenmiş bir kümesi olarak görülür. Fakat bu tablo belleğe yazıldıktan sonra, yalnız onu işleyen programın gereken şekilde değiştirilmesi şartı ile değişikliğe uğratabilir. Buradan görüldüğü gibi, aynı verilerin üzerinde değişik işlemler yapabilmek için onların temsili için öyle bir soyut matematik model seçilmelidir ki, veriler üzerinde yapılması gereken her bir işlemten önce uygun bir duruma getirilebilsin. Genellikle, söz konusu model olarak, küme ve kortej uygun değil. Çünkü birinci elemanların tekrarlanmasına, ikinci ise yer değişmesine imkan tanımamaktadır. Lakin bu iki kavramın arasında bulunan *Seri kavramı* söz konusu dönüşümler için çok müsaittir. Bir seride elemanların tekrarlanıp tekrarlanmaması ve yerlerinin değişip değişmemesi hiç de önemli değil. Bu yüzden, başlangıç veri yapısı olarak seri kabul edilirse, bu yapı istenildiği anda, çok kolay bir şekilde korteje, kümeye, tabloya, strüktüre vs. dönüştürülebilir. Görüldüğü gibi, bir seri, kendi elemanlarının yalnız fizik düzenini göstermekte, lojik düzen hakkında ise hiç bir bilgi bulundurmamaktadır. Bir seriye söz konusu türden olan lojik bilgi ilave etmekle, o lojik olarak istenilen yapı olarak sunulabilir. Fakat , bu durumda serinin matematik tabiatı değişeceği için uygun şekilde tanımlaması da gerekmektedir. Bu yüzden değişik işaretlerin kullanılmasına ihtiyaç vardır.

Bu çalışmada, gösterilen amaçla aşağıdaki işaretlemeler kullanılmıştır:

$\bar{S}$: Seri; $\hat{S}$: Küme; $\hat{\hat{S}}$: Multi-küme; $\bar{S}$: Kortej (Vektör)

VE'larının kendileri ile değerlerini bir birinden farklandırmak için, değer identifikatorlerinin üzerine nokta konulacaktır, mesela $\dot{s}$ gibi.

Veri sıkıştırma için gereken yöntemin seçilmesi, sıkıştırılması gereken verinin bulundurduğu fazlalık tipine, her bir fazlalık tipi ise belli sebeplere bağlıdır. Bu ise fazlalık tiplerini, onların oluşmasına yol açan sebeplere göre sınıflandırma imkanı sunmakta, fazlalık tiplerinin kesişmeyen kümelere bölünebilmesi gibi bir perspektife yol açmaktadır.

Veri tabanlarında bulunan fazlalıklar aşağıdaki 4 sınıfa bölünebilir [4,5,14,54,55]:

- I. Kavramsal fazlalık. Bu fazlalık, bazı veri birimlerini (eleman, kayıt, file vs.) bir kaç defa veri tabanına koymak mecburiyetinden kaynaklanmaktadır.
- II. İstatistiksel fazlalık. Bu fazlalık, veri tabanında literal (harf, rakam, noktalama işareti) ve literal gruplarının kullanılma sayılarının önemli bir oranda farklı olmasından kaynaklanmaktadır.
- III. Yapısal fazlalık. Bu fazlalık, bir takım verinin temsili için seçilmiş olan biçim ve yapılarının bu verilerin hepsi için aynı derecede tasarruflu olmamasından kaynaklanmaktadır.
- IV. Kod fazlalığı. Bu fazlalık, toplam sayısı 2^n den farklı olan literallerin veya veri elemanlarının eşit uzunluklu ikili sayılarla kodlanmasından kaynaklanmaktadır.

İlgili kaynaklarda gösterilmiş olduğu gibi, kavramsal fazlalıklar, veri tabanına, esneklik ve mimari amaçlarla koyulduğuna göre, ona dokunulmamalıdır. Fakat yapısal ve istatistiksel fazlalıklar veri tabanına bilerekten ve faydalı bir iş yapışları diye koyulmadığından, onların mümkün olduğu kadar giderilmesi gerekmektedir.

İstatistiksel fazlalık çok geniş ve detaylı bir şekilde araştırılmış ve onun giderilmesi için çok sayıda yöntem önerilmiştir. Bunlardan en isabetli olanları ve bu güne kadar geçerlilik ve güncelliklerini koruyanlar ise Shannon- Fano ve Huffman yöntemleridir [5,8,22,56]. Yapısal fazlalıklar genellikle yalnız belli veri yapılarına yönelik olarak

incelenmiş, genel olarak ise yok denilecek kadar az araştırılmıştır. Bu yüzden, bu tez çalışmasında yalnız yapısal fazlalıkların incelenmesi ve giderilmesi yolları ile ilgili olan problemler ele alınmıştır.

Yapısal enformasyon teorisi, kütüklerin ayrık (discrete) yapısını ele almakta ve onları sadece enformasyon elemanlarının sayılması yolu ile değerlendirmektedir [54]. Yapısal fazlalık ise değişik sebeplerden ve değişik şekillerde meydana çıktığı için yalnız bu teori çerçevesinde değerlendirilemez. Bu yüzden, söz konusu fazlalık tiplerinin araştırılması ve giderilmesi için özel yaklaşımlar gerekmektedir. Bu amaçla geliştirilmiş olan sıkıştırma yöntemleri aşağıdaki iki soyut dönüştürmeye dayanmaktadırlar:

$$F1: \bar{S} \rightarrow \bar{S}' \quad \text{ve} \quad F2: \bar{S} \rightarrow \bar{D}$$

Burada $\bar{S}$ verilerin başlangıç yapısı, $\bar{S}'$ ise verilerin sıkıştırmadan sonraki yapısı, $\bar{D}$ ise açma sürecinde $\bar{S}'$ yapısını $\bar{S}$ yapısına dönüştürmek için gereken yardımcı bilgidir. Halen (yıl 1983) çok sayıda veri sıkıştırma yöntemi bulunmaktadır [4,5,7-12,14,22,36,38-41,52,57-63]. Lakin bunların tümü veri tabanlarında kullanılmak için yararlı değil. Bu amaçla kullanılabilecek sıkıştırma yöntemleri ^{nir}belirleyici nitelikleri aşağıdakilerdir [12]:

1. Dönüşümlülük, yani sıkıştırılmış veriler hiç bir hataya yol verilmeden başlangıç durumlarına getirilebilmelidirler
2. Değerlendirme, yani sıkıştırma katsayısının ne kadar olabileceği ve gerçek uygulamalarda ne kadar olduğu değerlendirilebilmelidir
3. Pratik uygulamalar için önemli olacak kadar sıkıştırma yapabilmelidir
4. Universallık, yani sayı ve karakter tip verilere, yaklaşık olarak, aynı derecede başarı ile uygulanabilmelidir .

Ekonomik problemlere yönelik OKS 'lerde bulunan fazlalık tipleri [14,64,65]' de incelenmiş ve sınıflandırılmıştır. Veri tabanlarında uygulanması önerilmekte olan sıkıştırma yöntemleri ve onların sınıflandırılması ise [12,14]'de verilmiştir. Fakat önerilmekte olan yöntemlerin hiç biri veri tabanlarında bulunan yapısal fazlalıkların tümünü giderme imkanına sahip değil. Bunun esas sebebi, söz konusu yöntemlerin değişik fazlalık tipleri arasındaki ilişkileri dikkate almamasıdır.

Yukarıda gösterilmiş olan boşlukları doldurarak daha etkin olan sıkıştırma yöntemleri elde etmeği amaçlayan bu çalışmada, çözülmesi gereken problemler aşağıdaki gibi sıralanabilir:

- Yapısal fazlalık tiplerinin kaynaklarını araştırmakla onları meydana gelme sebeplerine göre sınıflandırmak
- Yaygın olan sıkıştırma yöntemlerini veri tabanına uygulanabilirlik açısından incelemek ve değerlendirmek
- Sıkıştırma yöntemlerinin ve fazlalık tiplerinin benzerlik oranını belirlemek, ve fazlalık tipleri ile sıkıştırma yöntemleri arasında var olan sıkı – sıkılgan ilişkilerini tespit etmek
- Sıkıştırma yöntemlerini fonksiyonel-eşdeğer sınıflara bölmek ve buna dayanarak bütün yapısal fazlalıkları giderecek olan minimum sayıda yöntemler kümesi belirlemek
- Mevcut yöntemlerin bıraktığı boşlukları dolduracak yeni yöntemler geliştirmek
- Her bir yöntemin kullandığı ve ürettiği veri yapısını belirlemek
- Verilerde bulunan fazlalık tiplerine göre sıkıştırma yöntemlerinin uygulama sırasını belirlemek

- Sıkıştırma yöntemlerinin ortak ve farklı yanlarını ortaya çıkarmakla modüler algoritmalar ve bu modüllerden gereken algoritmayı ve programı sentez edecek derleyici kurmak

1.3. Biçimsel fazlalıkların araştırılması ve giderilmesi yöntemleri

Genellikle, bir $\hat{S}$ kümesinin elemanlarını temsil etmek için uzunluğu $l(f)$ olan sabit bir biçim(format) kullanılmaktadır ki, bu da

$$\hat{S}_n = \{\dot{s}_i / \dot{s}_i \in \hat{S} \ \& \ l(\dot{s}_i) < l(f)\}$$

alt kümesinin elemanları için fazla olabilir. Burada $l(\dot{s}_i)$, $\dot{s}_i \in \hat{S}$ elemanının uzunluğudur. Biçimsel fazlalık olarak tanımlanan bu fazlalık tipi, U_{for} olarak işaretlenecektir. Örnek olarak, Tablo 1.1'in ikinci satırında birinci ve onuncudan başka tümü biçimsel fazlalığa sahip olan değerler gösterilmiştir. Böyle değerler, soluna ve sağına doldurucu semboller (sayısal değerler için 0, karakter tip değerler içinse boşluk işareti) eklemekle onlar için ayrılmış alanın biçimine kadar tamamlanırlar [5].

Tanım 1. Belirli bir değer için kullanılan doldurucular zinciri (DZ), art arda gelen sonlu sayıda doldurucu sembollerden (DS) oluşmaktadır.

Tanım 2. Bir değer kodunun sol sınırından başlayan DZ sol DZ, sağ sınırından başlayan DZ ise sağ DZ adlanır.

Tabii ki tam sayılar için yalnız sol DZ, kesir sayılar içinse yalnız sağ dizi kullanılabilir. Tam ve kesir kısımları olan sayılar ise bilgisayar dahilinde kayan noktalı sayılar olarak bir tam ve bir kesir olarak temsil edildiklerinden bunlar için DZ türü tanımlanmamıştır. Karakter tip değerlere gelince ise sola doğru hizalanmış olan değerler

için sağ DZ'ler, sağa doğru hizalanmış olan değerler içinse sol DZ'ler uygulanması gerekir.

Tablo 1.1. Verilerin ayrıştırıcı işaretler ve sınıflandırma diyagramları yöntemleri ile sıkılması

İ	$\dot{s}_i$	Ayrıştırıcı işaretler yöntemi (AİY)		Sınıflandırma diyagramları yöntemi (SDY)		
		$\bar{s}_i$	S_p	K_i	$\bar{s}_i$	$\bar{S}'_K$
1	1424	1424	1424*303*49*581*47*	4	1424	142430349581471015
2	0303	.303	101*593*483*35*1680* 79*141*101*32*67	3	.303	934833516807914110 13267
3	0049	..49		2	..49	
4	0581	.581		3	.581	
5	0047	..47		2	..47	
6	0101	.101		3	.101	
7	0593	.593		3	.593	
8	0483	.483		3	.483	
9	0035	..35		2	..35	
10	1680	1680		4	1680	
11	0079	..79		2	..79	
12	0141	.141		3	.141	
13	0101	.101		3	.101	
14	0032	..32		2	..32	
15	0067	..67		2	..67	

U_{for} tip fazlalık aşağıdaki yöntemlerden biri ile giderilir:

- Ayrıştırıcı işaretler yöntemi (AİY) [5,11,14,66]
- Sınıflandırıcı diyagramlar yöntemi (SDY) [14,65]

Bu yöntemlerin her ikisine de ait olan özellikler aşağıdakilerdir:

- Başlangıç veri yapısı olarak $\bar{S} = \{\dot{s}_i\}_{i=1,2,\dots,N}$ değerler serilerini kullanırlar
- Her bir $\dot{s}_i$ bir

$$\bar{s}_i = \langle a_1^i, a_2^i, \dots, a_j^i, \dots, a_n^i \rangle = \langle a_j^i \rangle_{j=1,2,\dots,n} \quad (1.1)$$

korteji olarak görülür

c) Her bir $\bar{s}_i = \langle a_j^i \rangle_{j=1,2,\dots,n}$ kortejinden uzunluğu r olan sol DZ'yi kesmekle

$$\bar{s}_i' = \langle a_j^i \rangle_{j=r+1,\dots,n} \quad (1.2)$$

kesilmiş korteji elde edilir.

Kesilmiş kortejler uygun değerin serideki sırası ile oluşturulur ve aynı sıra ile de yeni bir veri yapısında yerleştirirler. Böylece,

$$\bar{S} = \{\bar{s}_i\}_{i=1,2,\dots,N} = \{\{a_j^i\}_{j=1,2,\dots,n}\}_{i=1,2,\dots,N} \quad (1.3)$$

başlangıç veri yapısı

$$\bar{S}' = \{\bar{s}_i'\}_{i=1,2,\dots,N} = \{\{a_j^i\}_{j=r+1,\dots,n}\}_{i=1,2,\dots,N} \quad (1.4)$$

sıkıştırılmış veri yapısına çevrilmiş olur.

Ayrıştırıcı işaretler yöntemi(AİY)

AIY'nin özelliği, komşu kesilmiş kortejler arasına ayrıştırıcı semboller (mesela,*) koyulmasından ibarettir. Bu yöntem gereğince yardımcı bilgi D_y esas bilgi $\bar{S}'$ ile birlikte aşağıdaki veri yapısını oluşturmaktadır.

$$S_p = \langle \langle \bar{s}'_i, * \rangle \rangle_{i=1,2,\dots,N} \Leftrightarrow \langle \langle a_j^i \rangle \rangle_{j=r+1,\dots,n,} \rangle \rangle_{i=1,2,\dots,N} \quad (1.5)$$

Bu veri yapısının açılması, a_{r+1}^i bileşeninden başlayarak ve ardışık olarak gerçekleştirilir. i ve $i+1$ numaralı doldurucu semboller arasındaki bileşenler(rakamlar) $\bar{s}'_i$ kesilmiş korteji, yani $\bar{s}_i$ değerinin sıkıştırılmış şekli olarak algılanır. Örnek olarak, Tablo 1.1'de(2,3 ve 4. Sütunlar) 15 tane sayının AİY ile sıkıştırılması gösterilmiştir. Bu yöntem aşağıdaki niteliklere sahiptir:

1. Dönüştürümlüdür
2. Pratik uygulamalarda yeteri kadar etkilidir
3. Sayısal ve karakter tip verilere uygulanabilir
4. Sıkıştırılmış verilere yalnız ardışık ulaşım sağlayabilir
5. Bir verinin bozulması diğerlerinin doğru açılmasını engellemez
6. Sıkıştırılmış olan verilere gizlilik niteliği veremez

Sınıflandırıcı diyagramlar yöntemi(SDY)

SDY'ye göre her bir kesilmiş kortejin uzunluğu kayıt altında tutulmaktadır. Bu sebepten, her bir $\bar{S} = \{\bar{s}_i\}_{i=1,2,\dots,N}$ serisine karşılık olarak

$$S_k = \langle \bar{s}'_i \rangle_{i=1,2,\dots,N} \Leftrightarrow \langle a_j^i \rangle_{j=r+1,\dots,n,} \rangle \rangle_{i=1,2,\dots,N} \quad (1.6)$$

$$D_k = \langle K_i \rangle_{i=1,2,\dots,N} \quad (1.7)$$

veri yapılarını üretmektedir. Bu iki yapı birleştirildiğinde ise aşağıdaki tek yapı meydana gelmektedir.

$$S_k = \{ \langle \bar{s}_i' \rangle_{i=1,2,\dots,N}, \langle K_i \rangle_{i=1,2,\dots,N} \} = \{ \langle \langle a_j^i \rangle_{j=1,2,\dots,n} \rangle_{i=1,2,\dots,N} \}$$

Bu yapının kesin olarak açılabilmesi için sadece a_{r+1}^1 'in ve K_1 'in adresinin bilinmesi gerekir. Bu durumda, her hangi f numaralı kesilmiş kortejin birinci bileşeninin etkin adresi aşağıdaki gibi hesaplanır.

$$N_f = \sum_{i=1}^{f-1} K_i + 1$$

Böylece, $\bar{s}_i'$ kesilmiş kortejine ulaşabilmek için $\langle N_f, K_f \rangle$ çiftini kullanmak yeterli oluyor.

Yukarıda gösterildiği gibi, AİY' den farklı olarak, SDY her hangi bir kesilmiş korteye direkt ulaşma imkanı sağlamaktadır. Fakat bunun için yardımcı bilginin $\langle K_i \rangle_{i=1,2,\dots,f}$ kısmını seri işleme tabi tutmak gerekir. Örnek olarak, 15 tane sayının SDY ile sıkıştırılması Tablo 1.1' de gösterilmiştir (2,5, 6 ve 7. sütunlar).

SDY aşağıdaki niteliklere sahiptir

1. Dönüşümlüdür
2. Pratik uygulamalarda yeteri kadar etkilidir
3. Kesilmiş olan kortejlere direkt ulaşım sağlamaktadır
4. Sayısal ve karakter tip verilere aynı derecede başarı ile uygulanabilir
5. Bir elemanda olan bozukluk sonraki elemanların tümünün yanlış seçilmesine sebep olabilir. Çünkü, yardımcı bilginin herhangi bir elemanı bozulduğunda, uzunluğu o eleman tarafından belirlenen kesilmiş kortej eksik veya fazlası ile seçilecek, bu yanlışlık ise onu takip eden diğer kesilmiş kortejlerin seçilmesine de yansımacaktır.

6. Yardımcı bilgi olmadan sıkılmış olan verileri doğru bir şekilde açmak çok zordur.

Bu nitelik söz konusu yöntemle sıkıştırılmış olan verilere önemli bir ölçüde gizlilik kazandırabilir.

AİY ve SDY'nin karşılaştırılması

Bu yöntemlerin esas niteliklerinin mukayese edilmesinden görüldüğü gibi, AİY'inin SDY karşısındaki avantajı bir kortejde olan bozukluğun diğer kortejlerin doğru açılmasını etkilememesi, dezavantajı ise açma işleminin nispeten yavaş olmasıdır ki bu da her bir bileşenin ayrıştırıcı işaret olup olmadığını kontrol etmek mecburiyetinden kaynaklanmaktadır.

SDY'nin AİY karşısındaki avantajları daha hızlı olması ve gizlilik sağlaması, dezavantajı ise bir yerde olabilecek bozukluğu geriye kalan ne varsa hepsine yansıtmasıdır.

[14]'de verilen bilgiye göre, SDY sıkma ve açma modlarında, AİY'ye göre, uygun olarak 4-5 ve 2-3 kat daha hızlı çalışmaktadır. Bunun yanında, az sayıda veriler için SDY'nin sağladığı sıkıştırma katsayısı daha yüksektir. Bunun sebebi, SDY'nin kesilmiş kortejlerin ayrıştırılması için daha kısa kodlar kullanmasıdır. Fakat verilerin sayısı 128'i aştığında AİY daha yüksek sıkıştırma katsayısı sağlamaktadır. Çünkü bu durumlarda sınıflandırma diyagramının bir elemanı, AİY'nin kullandığı ayrıştırma işaretinden, bit sayısı olarak, daha uzun olan bir kod gerektirir. Ancak, ikinci ve üçüncü kuşak bilgisayarların sistem programları arasında yer alan sıkıştırma programları AİY kullandığı için [66,67] bu yöntem, pratik olarak daha geniş bir uygulama alanı bulmaktadır.

Bu çalışma için esas uygulama alanı olan sanayi müesseselerinin pasaportlarını içeren veri tabanı gizli olması gereken veriler içerdiği için, bu veri tabanında olan biçimsel fazlalıkları gidermek için kullanılabilecek tek araç SDY'dir.

1.4. Göreli fazlalıkların araştırılması ve giderilmesi yöntemleri

Bir değerler serisinin komşu elemanları arasındaki farkların uzunlukları elemanların kendi uzunluklarından kısa olduğunda göreli olarak tanımlanan fazlalık (U_{rel}) meydana çıkar. Bu tip fazlalık, aralarındaki farklar $D \leq Q^{l-2}$ bandında değişen düzenlenmiş değerler serilerinde meydana çıkar. Burada Q : sayı sisteminin tabanı, l ise bir değeri ifade etmek için kullanılan kodun uzunluğu olup, bütün değerler için aynı olan bir sayıdır. U_{rel} tip fazlalık aşağıdaki yöntemlerle giderilir:

- Farklar yöntemi [12,58,60,69,72]
- Artışlar yöntem [8,11,60,63]

Farklar Yöntemi

Farklar yöntemi(FY), $\bar{S} = \{\dot{s}_i\}_{i=1,2,\dots,N}$ serisi şeklinde verilmiş olan sayısal değerleri sıkıştırmak için kullanılmaktadır. FY'ye göre sıkıştırılmış değerler aşağıdaki yollardan biri ile elde edilir:

1. $\dot{s}_e = \min \{\dot{s}_i\}_{i=1,2,\dots,N}$; $\forall (i = 1,2,\dots,N)(\dot{s}'_i = \dot{s}_i - \dot{s}_e)$
2. $\dot{s}_e = \max \{\dot{s}_i\}_{i=1,2,\dots,N}$; $\forall (i = 1,2,\dots,N)(\dot{s}'_i = \dot{s}_e - \dot{s}_i)$
3. $\forall (i = 2,\dots,N)(\dot{s}'_i = \dot{s}_i - \dot{s}_{i-1})$
4. $\forall (i = 1,2,\dots,N-1)(\dot{s}'_i = \dot{s}_i - \dot{s}_{i+1})$

Bu yollarla elde edilmiş olan sıkıştırılmış veriler uygun olarak aşağıdaki yollarla açılır:

1. $\forall (i = 1,2,\dots,N)(\dot{s}_i = \dot{s}'_i + \dot{s}_e)$

2. $\forall(i = 1, 2, \dots, N)(\dot{s}_i = \dot{s}'_e - \dot{s}_i)$
3. $\forall(i = 2, \dots, N)(\dot{s}_i = \dot{s}'_i + \dot{s}_{i-1})$
4. $\forall(i = 1, 2, \dots, N-1)(\dot{s}_i = \dot{s}'_i + \dot{s}_{i+1})$

Bunların dışında FY'yi uygulamanın 5. bir yolu da mevcuttur [60]. Bu yol yalnız düzenlenmiş sayısal verilere uygulanabilir.

Söz konusu yola göre, $\bar{S} = \{\dot{s}_i\}_{i=1,2,\dots,N}$ serisi m tane parçaya ayrılır. Bu parçalardan her biri bir h sayısının misilleri olan bir sınırdan başlar. Bu durumda, her bir değer iki (i,j) indisle belirlenir. Birinci indis söz konusu değer hangi parçada olduğunu, ikinci indis ise söz konusu değer söz konusu parçanın hangi konumunda olduğunu gösterir. Bu yolla değerler aşağıdaki gibi sıkıştırılır.

$$\forall(j = 1, 2, \dots, G_m)(\dot{s}'_{mj} = \dot{s}_{mj} - H(j-1))$$

Burada G_m , m numaralı parçada bulunan değerlerin sayısıdır.

Bu yöntem küçük basamakları tutma yöntemi olarak tanımlanmakta ve yalnız düzenlenmiş sayısal değerler serisine uygulanabilmektedir.

Görüldüğü gibi FY'nin uygulanma yollarının tümü $\bar{S} = \{\dot{s}_i\}_{i=1,2,\dots,N}$ serisini $S_{PM} = \{\dot{s}'_i\}_{i=1,2,\dots,N}$ serisine çevirir. Bu durumda orijinal $\dot{s}_i$ ve sıkıştırılmış $\dot{s}'$ değerleri arasında aşağıdaki gibi bir ilişki meydana çıkıyor.

$$\forall(\dot{s}_i)((\dot{s}' \neq \dot{s}_i) \& (l(\dot{s}'_i) = l(\dot{s}_i)))$$

Yani, FY ile sıkıştırma sonucu biçim değişmediğinden, kısaltılmış değerlerin başına sıfırlar geldiği için hiçbir sıkıştırma da elde edilemez. Bu tespit aşağıdaki gibi şekillendirilebilir.

Tespit: FY hiçbir fazlalık gidermiyor. Sadece U_{rel} tip (görelî) fazlalığı U_{for} tip (biçimsel) fazlalığa çevirir.

Öneri: FY'nin uygulanması, yalnız ondan sonra U_{for} fazlalığını gideren bir yöntem uygulandığında faydalı olabilir.

Böylece, FY'yi kullanmakla sıkıştırma iki aşamada gerçekleştirilmelidir:

1. $\bar{S} = \{\dot{s}_i\}_{i=1,2,\dots,N} \longrightarrow S_{PM} = \{\dot{s}'_i\}_{i=1,2,\dots,N}$
2. 1. $S_{PM} = \{\dot{s}'_i\}_{i=1,2,\dots,N} \longrightarrow S_P = \langle\langle \dot{s}'', * \rangle\rangle_{i=1,2,\dots,N} (A\dot{I}Y)$ veya
- 2.2. $S_{PM} = \{\dot{s}'_i\}_{i=1,2,\dots,N} \longrightarrow S_P = \{\langle \dot{s}'', < K_i \rangle\}_{i=1,2,\dots,N} (SDY)$

FY aşağıdaki niteliklere sahiptir:

1. Dönüştümlüdür
2. Ancak AİY veya SDY ile birlikte etkin olabilir
3. Yalnız sayısal verilere uygulanabilir

FY'nin geriye kalan nitelikleri, ondan sonra uygulanacak olan yöntemin niteliklerine bağlıdır.

Artışlar Yöntemi

Artışlar yöntemi(A Y) hem sayısal, hem de karakter tip verilere uygulanabilir. Bu yöntemle göre $\bar{S} = \{\dot{s}_i\}_{i=1,2,\dots,N}$ değerler serisi $S_{PK} = \langle\langle \tilde{s}_{i-1}, \tilde{s}_i \rangle\rangle_{i=1,2,\dots,N}$ kartej çiftleri serisi olarak ele alınır. $\tilde{s}_{i-1}$ ve $\tilde{s}_i$ kartejlerinin bileşenleri soldan başlayarak karşılaştırılır

ve her iki kortejde de aynı olan bileşenler zinciri ikinci kortejden kesilir. Sonuçta her bir

$$< \bar{s}_{i-1}, \bar{s}_i > \text{ çifti bir } \bar{s}_i'$$

kesilmiş kortejine dönüştürülmüş ve bunların oluşturduğu

$$S_{PM} = \langle \langle \bar{s}', * \rangle \rangle_{i=1,2,\dots,N} \quad (1.9)$$

veri yapısı elde edilmiş olur.

(1.5) ve (1.9) yapılarının karşılaştırmasından görüldüğü gibi bu yapılar aynı olduğu için S_p ve S_{PM} veri yapıları aynı işlemlere tabi tutulabilir. Uygun açma düzeni yalnız onunla farklanır ki, kesilmiş olan basamakların yerine sıfırlar değil, bir önceki kortejin aynı basamaklarında bulunan semboller zinciri konulur [60].

Yukarıdaki açıklamalardan da görüldüğü gibi, AY aşağıdaki niteliklere sahiptir:

1. Dönüşümlüdür
2. Pratik uygulamalarda yeteri kadar etkindir [5,8]
3. Sayısal ve karakter tip bilgilere aynı başarı ile uygulanabilir
4. Kesilmiş kortejlere yalnız seri ulaşım sağlayabilir
5. Bir kortejin sıkılması ve ya açılmasında olan bozukluk sonraki kortejlerin tümüne yansır
6. Örnek değer (s_e) sıkılmış verilerden ayrı tutulduğunda bir miktar gizlilik sağlayabilir

Bu yöntem, F'Y'den farklı olarak:

- U_{rel} fazlalığı doğurmuyor ve bu yüzden sıkıştırmayı bir defada yapabilir
- Aritmetik işlemler gerektirmediğinden başka proseslerle paralel gerçekleştirilebilir

Fakat A'Y'e benzer olarak, bu yöntem de kesilmiş kortejlere direkt ulaşım sağlamıyor ve açma prosesi 2 aşamada gerçekleştirilir:

1. Kesilmiş kortejlerin seçilip ayrılması
2. Kesilmiş kortejlerin tam kortejlere tamamlanması

Ancak ilgili kaynaklarda gösterilenlerin aksine olarak, kortejlerin açılması sondan değil başlangıçtan başlamış olursa bu iki aşamayı bir aşama şeklinde gerçekleştirmek olur.

AY 2. ve 3. kuşak bilgisayarların sistem araçları arasında yer aldığı için geniş uygulama görmektedir. Bu yöntem, mikrokomutların adres kısımlarını sıkıştırmak, cihazların yazan veya gösteren organlarının koordinatlarını vermek vs amaçlarla kullanılmaktadır [41,43-48].

Farklar ve artışlar yöntemlerinin gerçekleştirilmesi örneği Tablo1.2'de verilmiştir.

Tablo 2.2. Verilerin Farklar ve artışlar yöntemleri ile sıkıştırılması

i	$\bar{s}_i$	Ayrıştırıcı işaretler kullanan farklar yöntemi		Artışlar Yöntemi	
		$\bar{s}'_i$	S_{PM}	$\bar{s}'_i$	S_{PM}
0	0000				
1	0032	0032	32*3*12*2*18*12*22**40	..32	32*5*47*9*67*79*101*
2	0035	0003	*162*180*98*12*831*156	...5	*41*303*483*581*93*1 424*680
3	0047	0012		..47	
4	0049	0002		...9	
5	0067	0018		..67	
6	0079	0012		..79	
7	0101	0022		.101	
8	0101	0000			
9	0141	0040		..41	
10	0303	0162		.303	
11	0483	0180		..483	
12	0581	0098		.581	
13	0593	0012		..93	
14	1424	0831		1424	
15	1680	0156		.680	

1.5. Serilerde elemanların tekrarlanmasından meydana gelen fazlalıkların araştırılması ve giderilmesi yöntemleri

Bu tip fazlalık (U_{TD}) OKS'lerin kullandığı veri kütüklerinin çoğunda bulunmaktadır. Bunlara en isabetli örnek olarak yeri boş bırakılmış çok sayıda veri içeren kütükler gösterilebilir[5,9,39,40]. Bu tip fazlalık aşağıdaki yöntemlerden biri ile giderilebilir:

- Mantık cetvellileri yöntemi (MCY) [5,9,59]
- Adres-Konum Yöntemi (AKY) [9,41]
- Kodlama Yöntemi ^{KY} [5,12]

MCY ve AKY yöntemleri

İlk iki yönteme has olan ortak özellik bundan ibarettir ki, en çok tekrarlanan $\dot{s}_e$ değeri $\bar{S} = \{\dot{s}_i\}_{i=1,2,\dots,N}$ serisinden seçilip ayrılır. Bundan sonra bu değere eşit olan bütün değerler seri içerisinde çıkarılır. Sonuçta $\bar{S}' = \{\dot{s}_i\}_{i \in \{1,2,\dots,N\}}$ sıkıştırılmış serisi elde edilmiş olur. Söz konusu yöntemler yalnız yardımcı bilginin yapısı açısından farklıdır.

MCY'ye göre, bir mantık cetveli kullanılır ve her bir $\dot{s}_i$ değeri için bu cetvelde, aşağıdaki gibi belirlenen, bir x_i biti tahsis edilir.

$$x_i = \begin{cases} 1, & \text{eğer } \dot{s}_i = \dot{s}_e \text{ ise} \\ 0, & \text{eğer } \dot{s}_i \neq \dot{s}_e \text{ ise} \end{cases}$$

Böylece, MCY ile sıkıştırılmış olan bir seri aşağıdaki yapıya sahip olur.

$$S_{MC} = \{\bar{S}', D_{MC}\} \quad (1.10)$$

AKY'ye göre ise yardımcı bilgi olarak aşağıdaki gibi indisler serisi oluşturulur.

$$D_{AK} = \{v_i\}_{i=1,2,\dots,N} \quad (1.11)$$

$$v_i = \begin{cases} \Delta, & \text{eğer } \dot{s}_i = \dot{s}_e \text{ ise} \\ i, & \text{eğer } \dot{s}_i \neq \dot{s}_e \text{ ise} \end{cases}$$

Bundan sonra Δ 'ye eşit olan bütün değerler D_{AK} serisinden çıkarılır ve sonuçta aşağıdaki sıkıştırılmış veri yapısı elde edilmiş olur.

$$S_{AK} = \{\bar{S}', D_{AK}\} \quad (1.12)$$

Açma sürecinde, pozisyonu D_{MC} 'de 1 veya D_{AK} 'de Δ olan değerlerin yerine $\dot{s}_e$ koyulur. Diğer değerler ise pozisyonlarına göre $\bar{S}'$ den alınırlar.

[9]'da gösterildiği gibi, AKY niteliklerinin tümüne göre MCY' den zayıf kaldığı için ileride dikkate alınmayacaktır.

Kodlama yöntemi

Kodlama yöntemi (KY), tekrarlanan elemanları başlangıç seriden kaldırmakla bir küme oluşturur ve bu kümenin elemanlarını kodluyor. Daha sonra başlangıç kümede

elemanların yerine onların kodlarını koyuyor. Neticede kodlar multi-kümesi elde edilmiş olur.

Bunun sonucunda $\hat{\bar{S}} = \{\hat{s}_i\}_{i=1,2,\dots,N}$ multi-kümesi ikili kodlar kümesine çevrilmiş olur. Formal olarak, bu çevirme aşağıdaki gibi ifade edilebilir.

$$K^+ : \hat{\bar{S}} \longrightarrow \bar{D}_K \quad (1.13)$$

$$\bar{D}_K = \langle d_i \rangle_{i=1,2,\dots,N} \quad (1.14)$$

Esas bilgi ise aşağıdaki kortejde toplanmış olur

$$\bar{S}_K = \{\bar{s}_j\}_{j=1,2,\dots,N_0} \quad (1.15)$$

Görüldüğü gibi, KY'ye ile sıkıştırılmış olan veri son olarak aşağıdaki yapıya çevrilir.

$$S_K = \{\bar{S}_K, \bar{D}_K\} \quad (1.16)$$

Açma sürecinde $\bar{D}_K$ kortejine $\hat{\bar{D}}_K$ multi-kümesi olarak bakılır ve $\bar{S}_K = \{\bar{s}_j\}_{j=1,2,\dots,N_0}$ korteji kullanılarak aşağıdaki dönüştürme gerçekleştirilir.

$$(K^+ : \hat{\bar{D}}_K \longrightarrow S_0) \longrightarrow \bar{S} \quad (1.17)$$

MCY ve KY aşağıdaki niteliklere sahiptir:

1. Dönüşümlüdür

2. Pratik uygulamalarda yeteri kadar etkilidir
3. Sayısal ve karakter tip verilere uygulanabilir
4. D_{MC} ve D_K kortejlerini seri şekilde işlemekle sıkılmış kortejlere direkt ulaşım sağlar
5. Bir sıkılmış kortejin bozulması diğerlerinin doğru açılmasını etkilemez

Eğer bir seride diğerlerine göre çok çok rastlanan bir değer varsa, o zaman MCY daha etkili olabilir. Bütün diğer durumlarda ise KY daha etkilidir.

[59]'da $\bar{S} = \{\dot{s}_i\}$ serisindeki elemanların tekrarlanma sayısının rasgele olduğu durum ele alınmıştır. MCY-M olarak adlandırılan bu yönteme göre, ilk seri 0. seri olarak tanımlanır,

$$S^0 = \{\dot{s}_i\}_{i=1,2,\dots,N} \quad (1.18)$$

S^0 için Y_1 mantık cetveli tertiplenir. Bu cetvelde en çok tekrarlanan $\dot{s}_e$ elemanına uygun konumlarda 1, diğerlerinde ise 0 yazılır ve 1 konumuna uygun olan değerler seriden çıkarılır. Bu prosedür son seride 1 eleman kalana kadar devam ettirilir. Eğer ilk seride bulunan orijinal elemanların sayısı m ise söz konusu prosedür m kere tekrarlanır. Ancak öyle durumlar olabilir ki, bu durumlardan sonra sıkıştırma mümkün olmuyor. Bu durumlar aşağıdaki şekilde tespit edilir. Eğer K . Seri elde edilmişse, $(K+1)$. serinin gereksizliği şu şarttan belirlenebilir.

$$\forall (i = 1, 2, \dots, N_k) \bar{\Xi}(\dot{s}_i) (I(\dot{s}_i) = \max \{I(\dot{s}_i)\}) \quad (1.19)$$

Sonuçta elde edilen sıkıştırılmış veri yapısı aşağıdaki gibi olur.

$$S_{MC-m} = \langle Y_1, \dot{s}_{e1}, \langle Y_2, \dot{s}_{e2}, \langle \dots \langle Y_m, \dot{s}_{em} \rangle \dots \rangle \rangle \rangle \quad (1.20)$$

Açma süreci $\langle Y_m, \dot{s}_{em}, S^m \rangle$ üçlüsünden başlıyor . Bu üçlünün açılması $\langle Y_{m-1}, \dot{s}_{em-1}, S^{m-1} \rangle$ üçlüsüne getirip çıkarır. Bu şekilde devam etmekle $\langle Y_1, \dot{s}_{e1}, S^1 \rangle$ üçlüsüne varılır ki bundan da S^0 serisine varılır.

Görüldüğü gibi, MCY-M sade MCY'nin çok kat olarak uygulanması yöntemidir. Bu yüzden, MCY' den bir az yüksek sıkıştırma kat sayısı sağlasa da çok yavaş çalışmaktadır.

Alt seriler yöntemi

U_{TD} fazlalığını gidermenin özel yollarından biri de bir seri içerisinde art arda tekrarlanan alt serileri tespit etmek ve onları belli bir yöntemle kodlamaktır [5,12,22]. Bu yöntemle göre her bir alt serinin birinci elemanı ve bu elemanın tekrarlanma sayısı gösterilir. [5,14]'de gösterilmiş olduğu gibi, bu yöntem genellikle, MCY yönteminden geri kalmaktadır. Fakat ilk seri az sayıda ve uzun olan alt serilerden oluştuğu durumlarda daha verimli olabilir.

MCY otomatlaştırılmış kontrol sistemlerinde geniş uygulama görmektedir. Bu yöntem çok sayıda eleman bulunduran veri kütüklerinin, anahtar matrislerinin, makine kelimelerinin vs. sıkıştırılması için ve en çok da ilişkisel veri tabanlarında kullanılmaktadır[5]. Örnek olarak, Tablo 1.3'de çelik markalarına ait olan verilerin [68] MCY ve KY yöntemleri ile sıkıştırılması gösterilmiştir .

Tablo 1.3. Verilerin MCY ve KY yöntemleri ile sıkıştırılması

No	$\bar{S}$	Mantık cetvelleri yöntemi (MCY)				Kodlama yöntemi(KY)		
		MCY-1, $\dot{s}_{e1} = 3PC$		MCY-2 $\dot{s}_{e2} = 08KP$				
		D_{MC1}	S^1	D_{MC2}	S^2	D_K	S_K	$\hat{\bar{D}}_K$
1	08KP	0	08KP	1	3KP	00	08KP	00
2	08KP	0	08KP	1	OPC	01	3PC	00
3	3PC	1	3KP	0		10	3KP	01

4	3PC	1	OPC	0		11	OPC	01
5	3PC	1						01
6	3KP	0						10
7	OPC	0						11
8	3PC	1						01
9	3PC	1						01
10	3PC	1						01

Tablo 1.3'den görüldüğü gibi, $\dot{s}_{e1} \cup \dot{s}_{e2} \cup S^2 = S_K$ olduğundan, MCY ve KY yalnız yardımcı bilginin yapısına göre farklılırlar. Buradan da görüldüğü gibi, bu yöntemler U_{TD} tip fazlalığı kaldırmakla az da olsa U_{rel} tip fazlalık üretebilirler.

1.6. Değişik serilerde aynı elemanların bulunmasından meydana gelen fazlalıkların araştırılması ve giderilmesi yöntemleri

Bu tip fazlalık (U_{OE}) birkaç kesişen seri olduğunda meydana çıkar ve aşağıdaki yöntemlerin biri ile gideriler:

- Kodlama yöntemi (KY)
- Bit matrisi yöntemi(BMY) [5]

Bu iki yöntemin her ikisinde de

$$\bar{S}_1 = \{\dot{s}_{1i}\}_{i=1,2,\dots,N_1}; \quad \bar{S}_2 = \{\dot{s}_{2i}\}_{i=1,2,\dots,N_2}; \quad \dots; \quad \bar{S}_m = \{\dot{s}_{mi}\}_{i=1,2,\dots,Nm}$$

serileri uygun olarak aşağıdaki kümeler olarak ele alınır:

$$\hat{S}_1 = \{\dot{s}_{1i}\}_{i=1,2,\dots,N_1}; \quad \hat{S}_2 = \{\dot{s}_{2i}\}_{i=1,2,\dots,N_2}; \quad \dots; \quad \hat{S}_m = \{\dot{s}_{mi}\}_{i=1,2,\dots,Nm}$$

ve bunların elemanlarından aşağıdaki küme oluşturulur.

$$\bar{S}_0 = \bigcup_{t=1}^m \bar{S}_t = \{\dot{s}_j\}_{j=1,2,\dots,N_0} \quad (1.22)$$

Böylece, her iki yöntem göre de sıkıştırılmış esas bilgi aynı şekilde oluyor. Fark ise yalnız yardımcı bilginin organizasyonundadır.

KY'ye göre sıkıştırılmış olan esas bilgi aşağıdaki şekilde elde edilir.

Her bir $\dot{s}_j \in \bar{S}_0$ elemanı uzunluğu $l(d_i) = \lceil \log_2 N_0 \rceil$ olan bir d_i kodu ile kodlanır ve bu yolla $\bar{D}_K = \{d_j\}_{j=1,2,\dots,N_0}$ kümesi elde edilir. Bundan sonra, her bir seri $\bar{\bar{S}}_t$ multi-kümesinin bir gerçekleşmesi olarak ele alınarak aşağıdaki dönüştürme yapılır.

$$K^+ : \bar{\bar{S}}_t \longrightarrow \bar{D}_K \quad (1.23)$$

Bu dönüştürme sayesinde sıkıştırılmış olan veri yapısı aşağıdaki gibi olur.

$$\bar{S}_K = \{\dot{s}_j\}_{j=1,2,\dots,N_0} \quad (1.24)$$

Bu yapıya yardımcı bilgi de eklendiğinde ise aşağıdaki veri yapısı elde edilir.

$$S_K = \{\bar{S}_K, \{D_{Kt}\}_{t=1,2,\dots,m}\} \quad (1.25)$$

(1.16) ve (1.25) veri yapılarının karşılaştırılmasından görüldüğü gibi, (1.16) yapısı (1.25) yapısının özel bir biçimidir. Yani, U_{TD} fazlalığı, U_{QE} fazlalığının bir seride meydana

çıkma şeklidir. Bu yüzden, daha kapsamlı olduğu için, ileride tek bir seri için kodlama yönteminin de (1.25) yapılı sıkıştırılmış veri ürettiği var sayılacaktır.

Bit matrisi yöntemine (BMV) göre sıkıştırmanın özelliği boyutları $m \times N_0$ olan bir matris kurmak ve bu matrisin satır ve sütunlarının kesişme noktalarındaki değerleri aşağıdaki şekilde belirlemektir:

$$C_{ii} = \begin{cases} 1, & \text{eğer } \dot{s}_i \in \bar{S}_i \\ 0, & \text{eğer } \dot{s}_i \notin \bar{S}_i \end{cases}$$

Bu durumda, her bir $i \in \{1, 2, \dots, N_0\}$ ile $\dot{s}_i \in \bar{S}_0$ arasında olan ilişkinin korunabilmesi için $\bar{S}_0$ kümesinin $\bar{S}_0 = \langle \dot{s}_i \rangle_{i=1, 2, \dots, N_0}$ korteji olduğu var sayılır. Böylece, BMV'nin $\{\bar{S}_i\}_{i=1, 2, \dots, m}$ seriler kümesine uygulanması, aşağıdaki veri yapısına getirip çıkarır.

$$S_{BM} = \{\bar{S}_0, M(m \times N_0)\} = \{\bar{S}_0, \{C_i\}_{i=1, 2, \dots, m}\} \quad (1.26)$$

Burada, C_i , M matrisinin bir satırıdır.

Açma sürecinde istenilen $t \in \{1, 2, \dots, m\}$ için $\{\bar{S}_0, C_t\}$ çifti kullanılarak $\bar{S}_t$ serisi aşağıdaki gibi elde edilir.

$$\bar{S}_t = \{\dot{s}_i / C_{it} = 1\} \quad (1.27)$$

Fakat, söz konusu yöntemle göre bir kaç seri aynı bir sıkıştırma işlemine tabi tutulduğundan, elemanların seriler içerisindeki düzeni korunamaz. Bu yüzden söz konusu yöntem, yalnız elemanların seriler içerisinde yerleşme düzeni önemli olmadığıda uygulanabilir.

BMY, U_{OE} fazlalığının yüksek olduğu ortak elemanlar bloklarına sahip olan listelerin sıkılmasında çok yaygın olarak kullanılmaktadır [5]. Örnek olarak, Tablo 1.4 de 3 serinin KY ve BMY ile sıkıştırılması gösterilmiştir.

Tablo1.4. Serilerin BMY ve KY yöntemleri ile sıkıştırılması

İlk seriler			S_0	Bit matrisi yöntemi(BMY)			Kodlama yöntemi(KY)			
$\bar{S}_1$	$\bar{S}_2$	$\bar{S}_3$		C_1	C_2	C_3	D_K	D_{K1}	D_{K2}	D_{K3}
350	350	350	7	1	1	1	0001	1000	1000	1000
7	7	7	80	0	1	0	0010	0001	0001	0001
85	93	90	85	1	0	0	0011	0011	0101	0100
605	620	660	90	0	0	1	0100	1001	1010	1011
170	170	170	93	1	1	0	0101	0111	0111	0111
93	80	100	100	0	0	1	0110	0101	0010	0110
			170	1	1	0	0111			
			350	1	1	1	1000			
			605	1	0	0	1001			
			620	0	1	0	1010			
			660	0	0	1	1011			

1.7. Aritmetik fazlalıkların araştırılması ve giderilmesi yöntemleri

Bu tip fazlalık (U_{Ar}) veri tabanlarında bir niteliğin(attribute) değerleri başka niteliklerin değerleri üzerinden hesaplanabilir halde, söz konusu değerlerin veri tabanına

konulmasından meydana gelir. Mesela, Sanayi müesseselerinin pasaportları aşağıdaki veri grubunu içermektedir[73]:

$$S = \{S_j, S_{j+1}, S_{j+2}\} :: = \{<plan>, <yerine getirilmesi>, <yerine getirme yüzdesi>\}$$

Burada S'in her bir elemanı uzunluğu N olan bir seridir. Bu yapıdaki bilginin tümünü korumak için sadece iki serinin korunmasına gerek vardır. Çünkü üçüncü seri, diğer ikisine göre hesaplanabilir. Fonların kullanımı dosyası da aynan bu yolla sıkıştırılabilir [73].

Aritmetik yöntem, sanayi müesseselerinin pasaportlarının otomatik işlenmesi sisteminde geniş şekilde uygulanmış ve %20 -%30 arasında bir sıkıştırma elde edilmiştir [73].

Bu yöntem aşağıdaki niteliklere sahiptir:

1. Dönüşümlüdür
2. Yeteri kadar etkilidir
3. Yalnız sayısal verilere uygulanabilir
4. Hiç bir diğer fazlalık türü türetmez

Bazı durumlarda, verilerde bulunmakta olan başka fazlalık tiplerinden dolayı, ArY'den sonra başka yöntemler de kullanılabilir.

1.8. Veri sıkıştırma ile veri yapıları arasında olan ilişkiler.

Çoğu durumlarda, verilerin ayrı ayrı bileşenleri arasında yapısal ilişkiler bulunur. Bu ilişkiler, veri bileşenleri ile birlikte *Veri Yapıları* olarak tanımlanırlar [4].

Bilgi işlem teknolojisinde çok sayıda veri yapısı kullanılmaktadır [5,9,53,74-76]. Lakin (1.3)-(1.7) ifadelerinden de görüldüğü gibi, veri sıkıştırma yöntemlerinin çoğu yalnız değerler

serileri ile çalışırlar. Bu yüzden, sıkıştırma yöntemlerini değişik veri yapılarına uygulayabilmek için, onların seri şekle getirilip getirilemeyeceğini bilmek gerekir.

Değişik veri yapılarının umumi modeli aşağıda gösterilen soyut modeldir [77].

$$S = \{\hat{E}, \hat{C}\} \quad (1.28)$$

Burada S veri yapısı, $\hat{E}$: veri elemanları(bileşenleri) kümesi, $\hat{C}$: elemanlar arasındaki ilişkiler kümesidir. Genellikle,

$$\hat{E} = \{z_i\}_{i=1,2,\dots,N} \quad \text{ve} \quad \hat{C} = \{c_{ij}\}_{i=1,2,\dots,N} \}_{j=1,2,\dots,N}$$

Burada $z_i \in \hat{E}$ veri elemanı, $c_{ij} \in \{0,1\}$ ikili ilişkidir. Halen (yıl 1983) en yoğun olarak kullanılan veri yapıları ise aşağıdakilerdir:

1. *Elemanları kayıtlar olan seri (ardışık) yapı*. Bu yapının elemanları, bellekte ciddi bir şekilde bir birinin ardınca yerleştirirler, yani $c_{ij} = 1 \leftrightarrow j = i+1$. Bu yüzden z_i ve z_{i+1} elemanları arasındaki ilişkileri göstermeğe lüzum kalmıyor. Başka bir deyişle ardışık veri yapısı, sadece bir seri olarak gösterilebilir. Yani,

$$\bar{S} = \{z_i\}_{i=1,2,\dots,N} \quad (1.29)$$

2. *Liste şekilli yapı* . Bu yapıda elemanlar arasında fiziksel düzen belirleyen bir ilişkinin vacipliği söz konusu değildir. Ancak burada var olan *Önceki-Sonraki* ilişkileri göstericiler (ilişki adresleri) yardımı ile verilir. Liste şekilli yapıda iki tür eleman kullanılabilir:

$\hat{E}$ kümesinin elemanlarını mantıki sıra ile ele alabilmek için $\hat{E}$ ve $\bar{C}$ arasında $F: \bar{C} \longrightarrow \hat{E}$ fonksiyonel ilişkisi kurulur. Eğer listede değerleri aynı olan birkaç eleman varsa, o zaman $|\{z_i\}_{i=1,2,\dots,N'}| = N'$, $|\hat{E}| = N$ ve $N' < N$ olacağı için, esas ve yardımcı (ilişki) bilgilerin ayrılıkta tutulması daha yüksek sıkıştırma katsayısı sağlayabilir.

Çift yönlü listeler de aynı şekilde sıkıştırılabilir. Burada sadece her bir eleman için iki gösterici kullanılmaktadır, yani

$$\bar{S}_{C2} = \{ \langle z_i, c_{i1}, c_{i2} \rangle \}_{i=1,2,\dots,N} = \{ E_i^{C2} \}_{i=1,2,\dots,N} \quad (1.32)$$

Esas ve ilişki bilgileri ayrı ayrı tutulduğunda bir değerler kümesi, bir ileriye doğru ilişkiler korteji, bir de geriye doğru ilişkiler korteji verilir. Yani,

$$S = \{ \hat{E}, \bar{C}_1, \bar{C}_2 \} = \{ \{ \{ z_i \}_{i=1,2,\dots,N'} \}, \langle C_{i1} \rangle_{i=1,2,\dots,N}, \langle C_{i2} \rangle_{i=1,2,\dots,N} \} \quad (1.33)$$

Bir ve iki istikametli listeler tek bir anahtara göre arama yapma imkanı sağlarlar. Lakin, genellikle kayıtlar birkaç arama anahtarı bulundurulur [5,9]. Bu durumda, aynı kayıtların birkaç defa tekrarlanması yerine kayıtların bir kümesi oluşturulur ve geriye kalan ne varsa ilişkiler üzerine koyulur. Yani,

$$S = \{ \hat{E}, \bar{C}_1, \bar{C}_2, \dots, \bar{C}_n \}; \quad \hat{E} = \{ z_i \}_{i=1,2,\dots,N'}; \quad \bar{C}_t = \langle c_{it} \rangle_{i=1,2,\dots,N_t} \quad (1.34)$$

Bu durumda $\forall (t, k \in \{1, 2, \dots, n\}) (N_t = N_k)$ şartı önemli değil. Çünkü, değişik listeler değişik sayıda kayıt içerebilir. Ancak, çok listeli yapı, çok büyük bir sıkıştırma katsayısı sağlayabilir.

3. *Ağaç yapısı* . Bu yapıda, her bir düğümünden (kaytan) sonra birkaç başka düğüm gelebilir. Fakat her bir kayıttan önce yalnız bir düğüm olabilir. Başlangıç düğüm ağacın kökü olarak tanımlanır. Ağacın kökünden çıkan dallar düğümlere, bu düğümlerden çıkan dallar da başka düğümlere vs. gider. Bu yapı bir biri ile hiyerarşik şekilde bağlanmış olan kayıtlar esasında veri tabanı kurmak için çok elverişlidir. Bu çalışma için esas teşkil eden sanayi müesseselerinin pasaportlarını saklama ve işleme sistemi de aynı veri yapısına sahiptir. Burada ağaç yapısına çok uygun olan “ Bakanlık-Müessese”, “Müessese-Mamul”, vs. gibi ilişkiler vardır [73,78]. Bundan başka bir bakanlığa çok sayıda müessese bağlıdır, her bir müessese ise çok sayıda memul üretmektedir. Bu durumda, her bir düğüm, yeteri kadar büyük olan ve çok sayıda anahtar bulunduran iki boyutlu bir kütüktür. Bu kütükler içerisindeki kayıtlar ise kendi aralarında değişik şekillerde ilişkide bulunmaktadır.

Halen (yıl 1983) ağaç yapılı verilerin olduğu şekilde belleğin lineer yapısına aktarılması yolu belli değil. Bu yüzden, ağaç yapısı önce liste yapısına dönüştürülür, sonra ise belli yollarla belleğe aktarılır. Bunun için aşağıdaki araçlardan yararlanılır [5,53]:

- Verilerin ardışık fizik modeli
- Göstericiler
- Referans listeleri
- Boole Matrisleri

4. *Şebeke yapısı*. Bu, elemanları arasında “ Önceki-Sonraki” ilişkileri olan bir yapıdır. Ağaç yapısından tek farkı, her bir sonrakinin birkaç öncekiye sahip olabilmesidir. Yani şebeke, elemanları arasında her tür ilişki olabilen bir yapıdır. Mesela, sanayi müesseselerinin pasaportları veri tabanında “Müessese” dosyası “Bakanlık”, “Bölge” ve “Arazi” dosyaları altındadır. İlave olarak, burada bilgi – referans dosyaları da vardır ki, arama şartlarına bağlı

olarak bir durumda “önceki”, başka durumda “sonraki” olarak kullanılabilir. Veri tabanlarında, şebeke yapısının her bir düğümünde istenildiği kadar büyük olabilen iki boyutlu bir dosya bulunur. Bu dosyada bulunan kayıtlar da bir biri ile göstericilerle ilişkilendirilirler. Bu yüzden, şebeke yapısı da aynen ağaç yapısı gibi lineer bellek modelin² aktarılabilir [5].

5. *Tablo yapısı*. Bu ikiboyutlu bir yapıdır. Tablonun her bir satırı bir kayıttan ibarettir. Tablo, ilişkisel veri tabanının yapısal temelini oluşturmaktadır. Bunun sebebi, istenilen veri yapısının normalleştirme yöntemleri ile tabloya çevrilebilmesidir. Tablo yapısı açısından, her bir veri tabanına üzerinde seçme ve birleştirme işlemleri yapılabilen tablolar topluluğu olarak bakılabilir. Tablolar, veri tabanında satırlar (kortejler) veya sütunlar (nitelikler) dizileri olarak tutulurlar. Bu durumda veri tabanı değişen şartlara daha iyi uyum sağlayabilir. Her iki halde de tablo yapısı bellek yapısına seriler olarak yansıtılır [79].

Satırlar ve ^{sütunlar} ayrı ayrı tutulduğunda uygun olarak aşağıdaki yapılar oluşur:

$$\bar{S}_K = \{z_i\}_{i=1,2,\dots,I} \quad (1.36)$$

$$\bar{S}_D = \{R_j\}_{j=1,2,\dots,J} \quad (1.37)$$

Burada z_i bir kortej, R_j ise bir niteliğin değerlerinden oluşan bir domendir.

Değişik nitelikler arasındaki ilişkiler ise ilişki referansında gösterilir.

$$S_{DI} = \{C_j\}_{j=1,2,\dots,J} \quad (1.38)$$

Burada C_j , R_j niteliği için referanstır.

Her bir domen, diğer domenlerle ilişkide olduğu için referanslar aşağıdaki gibi bir tablo oluştururlar.

$$C = \{ \{ C_{ik}^j \}_{i=1,2,\dots,J_j} \}_{K=1,2,\dots,J_j} \quad (1.39)$$

Burada J_j , R_j domenindeki elemanların sayısı; J_j , R_j niteliği ile ilişkide olan diğer niteliklerin sayısı; C_{ik}^j , R_K niteliğinin elemanı olup, R_j niteliğinin i numaralı elemanı ile ilişkide olan elemanın numarasıdır. Böylece, nitelikleri ayrı ayrı saklanılan ilişkiyel veri tabanı aşağıdaki şekilde ifade edilebilir.

$$S_T = \{ \{ R_j \}_{j=1,2,\dots,J}, \{ C_j \}_{j=1,2,\dots,J} \} \quad (1.40)$$

(1.29)-(1.40)' tan görüldüğü gibi, ele alınan veri yapılarının tümü, değişik seriler olarak ifade edilir ve bu seriler arasındaki ilişkiler onları işleyen programlar tarafından belirlenir ve kullanılır. Bu demektir ki, istenilen tip veri yapısı, belleğe aktarılmadan önce, seriler topluluğu şekline getirilmektedir ki, bu da sıkıştırma yöntemlerinin uygulanması için en elverişli veri yapısıdır. Fakat, verilerin mantıki yapılarının fizik yapılarına aktarılması sürecinde sıkıştırma yöntemlerinin yeri şu ana kadar (yıl 1983) yeteri kadar araştırılmamış olduğundan, sıkıştırma yöntemlerinin etkili bir şekilde uygulanması yolunda bazı zorluklar çıkmaktadır. Bu yüzden, aşağıda bu konu ele alınacaktır.

1.9. Verilerin lojik ve fizik yapılarının karşılıklı dönüşüm proseslerinin kısa analizi

Önceki paragraftan da görüldüğü gibi, OKS'de kullanılan veri yapıları direkt olarak belleğe aktarma açısından uygun değil [11]. Bu yüzden, genellikle seriler topluluğu olarak ifade

edilebilen bu lojik yapıları belleğe aktarmadan önce uygun dönüşümlere tabi tutmak gerekir [77]., Yani,

$$F_{C1}:S_L \longrightarrow S_A$$

Burada S_L : verilmiş mantıki veri yapısı, S_A ise aralık (ardışık, listesel, yığın, ağaç, şebeke, tablo vs.) bir veri yapısıdır.

Yukarıda gösterildiği gibi, S_A yapısı da S_L yapısı gibi direkt olarak belleğe aktarılamıyor. Bu yüzden, ikinci bir

$$F_{C2}:S_A \longrightarrow H_A$$

yapısal dönüştürme ile S_A yapısı $H_A = \{\bar{S}_1, \bar{S}_2, \dots, \bar{S}_i, \dots, \bar{S}_m\}$ serileri şeklinde gösterilir.

Belli olduğu gibi, H_A yığını belleğe aktarıldığında sadece her biri V sayıda bayt(byte) içeren bloklar ardışıklığı olarak ele alınır. Bu durumda lojik bloklar fizik bloklara (M_i) aşağıdaki şekilde yansıtılır.

$$F_{C3}: H_A \longrightarrow M \equiv H_A = \{\bar{S}_1, \bar{S}_2, \dots, \bar{S}_i, \dots, \bar{S}_m\} \xrightarrow{V} \{M_1, M_2, \dots, M_i, \dots, M_Q\}$$

Burada Q , fizik blokların sayısıdır. Böylece, verilerin lojik yapısının, fizik yapısına çevrilmesi problemi aşağıdaki 3 dönüşümle gerçekleştirilir:

$$1) F_{C1}:S_L \longrightarrow S_A$$

$$2) F_{C2}:S_A \longrightarrow H_A$$

$$3) F_{C3}:H_A \longrightarrow M$$

Bu dönüşümler aşağıdaki kompozisyon olarak uygulanırlar.

$$R^+ = F_{C3}^+ : (F_{C2}^+ : (F_{C1}^+ : S_L \longrightarrow S_A) \longrightarrow H_A) \longrightarrow M$$

Gerektiğinde, fizik yapı-lojik yapı dönüştürmesi ise aşağıdaki gibi gerçekleştirilir.

$$R^- = F_{C1}^- : (F_{C2}^- : (F_{C3}^- : M \longrightarrow H_A) \longrightarrow S_A) \longrightarrow S_L$$

Veriler sıkıştırıldığında, bu sisteme dördüncü bir F_{CS} dönüştürmesi de eklenir. Bu dönüştürme serilere uygulandığından dolayı F_{C2} dönüştürmesinden sonra uygulanması gerekir. Böylece lojik yapının sıkıştırma ile fizik yapıya ve sıkıştırılmış veri tutan fizik yapının lojik yapıya dönüştürülmesi aşağıdaki kompozisyonlar vasıtası ile gerçekleştirilir.

$$R^+ = F_{C3}^+ : (F_{C2}^+ : (F_{CS}^+ (F_{C1}^+ : S_L \longrightarrow S_A) \longrightarrow H_A) \longrightarrow H_S) \longrightarrow M$$

$$R^- = F_{C1}^- : (F_{C2}^- : F_{CS}^- (F_{C3}^- : M \longrightarrow H_A) \longrightarrow H_S) \longrightarrow S_A) \longrightarrow S_L$$

Burada, F_{CS}^+, F_{CS}^- , uygun olarak, sıkıştırma ve açma fonksiyonları H_S ise sıkıştırılmış veri yapısıdır.

$\bar{S}_1, \bar{S}_2, \dots, \bar{S}_i, \dots, \bar{S}_m$ serileri çok büyük olduğunda onların teker teker sıkıştırılması da

etkili olabilir. Bu durumda F_{C2}^+, F_{C2}^- çevirmelerine de gerek kalmaz.

I. Bölümün sonuçları

1. OKS'lerde kullanılan veri kütüklerinin çok büyük olduğu, yüksek oranda fazlalık içerdiği ve bu yüzden belleğin çok düşük bir verimle kullanılmakta olduğu ve bu problemin esas çözüm yollarından birinin veri sıkıştırması olduğu gösterilmiştir. Bu yaklaşım, özellikle büyük sanayi birleşmeleri, bölgeler, araziler ve benzer oluşumların bilgi işlem merkezlerinin normal faaliyeti açısından kaçınılmaz bir hale gelmiştir. Çünkü, söz konusu organizasyonda, bu günün(yıl 1983) teknolojisinin kaldıramayacağı kadar büyük çapta verilerin saklanması, işlenmesi ve iletilmesi gerekmektedir. Bu yüzden, veri sıkıştırma yöntemleri ele alınmış, bunların içerisinde bu gün için problem olan yapısal fazlalıkları gideren yöntemler seçilmiş ve her bir yöntem bütün detayları ile araştırılıp değerlendirilmiştir. Buna dayanarak, veri tabanlarında bulunan yapısal fazlalıkları giderme yöntemlerine karşı koyulması gereken istekler belirlenmiştir.
2. Değişik yapısal fazlalık tiplerinin meydana çıkma sebepleri belirlenmiş, bunları giderebilecek yöntemler araştırılmış, bu yöntemlerin kullandığı ve ürettiği veri yapıları ve onların soyut matematik modelleri belirlenmiştir ki bu da sıkıştırma yöntemleri arasında bir düzen olduğunu göstermiş ve bu yöntemlerin uygulanma sırasının belirlenmesi için bir teorik temel oluşturulmuştur.
3. Veri tabanlarında bulunan fazlalıkların 4 esas sınıfa bölüldüğü gösterilmiş, bunlardan yalnız en az araştırılmış olan yapısal fazlalıklar üzerinde durulmuştur. Burada: biçimsel, görelî, aritmetik, tekrarlanma vs. tip fazlalıklar ve bunları giderecek olan sıkıştırma yöntemleri derinden araştırılmıştır. Özellikle, bazı yöntemlerin bir tip fazlalığı gidermekle başka tip fazlalık ürettiği ve bu yüzden tek başına etkili olamayacağı gösterilmiş, yöntemler arasında bir ilişki kurmak için giderilen fazlalık- üretilen fazlalık ilişkileri detaylı olarak incelenmiştir. Bu ise bir veri yapısına art arda olarak birkaç

sıkıştırma yöntemi uygulamakla mümkün olduğu kadar yüksek sıkıştırma katsayısı elde etme yolunu açmıştır.

4. Yaygın olarak kullanılan soyut veri yapıları incelenmiş ve onlarda bulunabilecek fazlalık tipleri belirlenmiştir. Değişik tip veri yapılarının tümünün seriler şekline getirildiğini ve yalnız bundan sonra lineer bellek modeline aktarılabilirdiği gösterilmiş ve bu yüzden sıkıştırma yöntemlerinin pratik olarak soyut veri yapılarına bağlı olmadığını tespitine varılmıştır. Burada bu gün (yıl 1983) en güncel olan ilişkisel veri tabanının temelini oluşturan tablo yapılarına özel bir ağırlık verilmiştir.
5. Verilerin lojik ve fizik yapılarının karşılıklı dönüşümleri ele alınmış ve sıkıştırma yöntemlerinin bu süreçteki yeri belirlenmiştir. Bazı fazlalık tiplerinin her birinin birkaç yöntemle giderilebileceği, bazı yöntemlerin ise her birinin birkaç tip fazlalığı giderebileceği gösterilmiştir. Bu ise sıkıştırma yöntemlerini eşdeğer sınıflara bölmek ve bunlar arasından OKS'erde bulunan fazlalık tiplerinin tümünü giderebilecek sıkıştırma yöntemlerinin minimum kümesini bulmak problemini meydana çıkarmıştır.

2. VERİ SIKIŞTIRMA YÖNTEMLERİNİN FONKSİYONEL TAM KÜMESİNİN SEÇİLMESİ

2.1. Veri sıkıştırma yöntemlerinin karşılaştırmalı analizi

Veri sıkıştırma ile ilgili olan çalışmalarda ya tek bir yöntem, yada birkaç yöntem birlikte ele alınmaktadır [5,8,12,60]. Lakin yalnız [12, 64]'de değişik yöntemlerin aynı bir veri kütüğüne uygulanması sırası ele alınmıştır. Söz konusu sıra, aşağıdaki açılardan önemli olmaktadır:

1. Bazı yöntemler bir fazlalık tipini gidererek başka bir fazlalık tipi üretirler. Yani bir fazlalık tipini başka bir fazlalık tipine çevirirler. Bu ise sıkıştırma yöntemleri arasında bir uygulama sırasının olması gerektiğini göstermektedir.
2. Birkaç fazlalık tipi bulunduran veri kütüğüne öyle sıkıştırma yöntemlerinin ve öyle bir sıra ile uygulanması gerekir ki, onlardan her birinin ürettiği veri yapısı, en az diğer biri tarafından giriş veri yapısı olarak kullanılabilsin.

Veri sıkıştırma yöntemlerinin uygulanma sırası olarak [12, 64]'de teklif edilmiş olan usul, sınırlı tip veri yapılarına yönelik olması ve karmaşıklığı ile dikkati çekmektedir. Fakat yukarıda gösterilmiş olduğu gibi, söz konusu usulün en az yaygın olarak kullanılmakta olan veri yapılarının tümüne uygulanabilir olması ve yöntemlerin uygulama sırası açısından ciddi olması gerekir.

Tablo 2.1'de sıkıştırma yöntemlerinin bölüm I'de belirlenmiş olan esas karakteristikleri gösterilmiştir. Bu tabloda görüldüğü gibi, aritmetik yöntem (ArtY) ve bit matrisi yöntemi(BMY)'den başka bütün yöntemler giriş veri yapısı olarak seri kullanır, çıkış veri yapısı olarak da kortej üretir. Böylece, A_{vt} tarafından üretilmiş olan veri yapısı,

direkt olarak diğer yöntemler tarafından kullanılabilir. Fakat diğer yöntemlerin ürettiği veri yapısı $A_{rt}Y$ tarafından kullanılamaz. Böylece,

$$\forall (a \in H_5)(A_{rt}Y < a) \quad (2.1)$$

kuralı meydana çıkıyor ki bu da eğer aritmetik fazlalık varsa, ilk uygulananın AY olması gerektiği demektir. Burada $H_5 = \{SDY, AIY, AY, FY, MCY, KY, BMY\}$, 5 indisi ise sonraki açıklamaların rahatlığı için kullanılmıştır.

Tablo 2.1. Veri sıkıştırma yöntemlerinin esas karakteristikleri

Sıkıştırma yöntemi	Veri Yapısı		Fazlalık Tipi	
	Kullanılan	Üretilen	Giderilen	Üretilen
SDY	$S = \{\dot{s}\}_{i=1,2,\dots,N}$	$\{< \bar{s}'_i >_{i=1,2,\dots,N}, < K_i >_{i=1,2,\dots,N}\}$	U_{for}	-
AIY	$S = \{\dot{s}\}_{i=1,2,\dots,N}$	$<< \bar{s}'_i, * >>_{i=1,2,\dots,N}$	U_{for}	
AY	$S = \{\dot{s}\}_{i=1,2,\dots,N}$	$<< \bar{s}'_i, * >>_{i=1,2,\dots,N}$	U_{rel}	
FM	$S = \{\dot{s}\}_{i=1,2,\dots,N}$	$< \bar{s}' >_{i=1,2,\dots,N}$	U_{rel}	U_{for}
MCY	$S = \{\dot{s}\}_{i=1,2,\dots,N}$	$\{\bar{S}_L, \bar{D}_L\}$	U_{TD}	U_{rel}
KY	$S = \{\bar{S}_i\}_{i=1,2,\dots,m}$	$\{\bar{S}_{PK}, \{\bar{D}_{pkt}\}_{i=1,2,\dots,m}\}$	U_{OE}, U_{TD}	
BMY	$\bar{S}_i = \{\dot{s}_{ti}\}_{i=1,2,\dots,N_i}$	$\{\bar{S}_{BM}, \{\bar{D}_{Li}\}_{i=1,2,\dots,m}\}$		
$A_{rt}Y$	$S = \{\bar{S}_i\}_{i=1,2,\dots,m}$ $\bar{S}_i = \{\dot{s}_{ti}\}_{i=1,2,\dots,N_i}$	$\{\bar{S}_i\}_{i=1,2,\dots,m}$	U_{art}	-

Açıklama: SDY- Sınıflandırma diyagramı yöntemi; AIY- Ayrıştırıcı işaretler yöntemi; AY-Artışlar yöntemi; FM- Farklar yöntemi; MCY- Mantık cetvelleri yöntemi; KY-Kodlama yöntemi; BMY-Bit matrisi yöntemi; $A_{rt}Y$ - Aritmetik yöntem, U_{for} -Biçimsel fazlalık, U_{rel} -Görelî fazlalık, U_{TD} - Tekrarlanan değerler fazlalığı; U_{OE} : Ortak eleman fazlalığı, U_{art} - Aritmetik fazlalık.

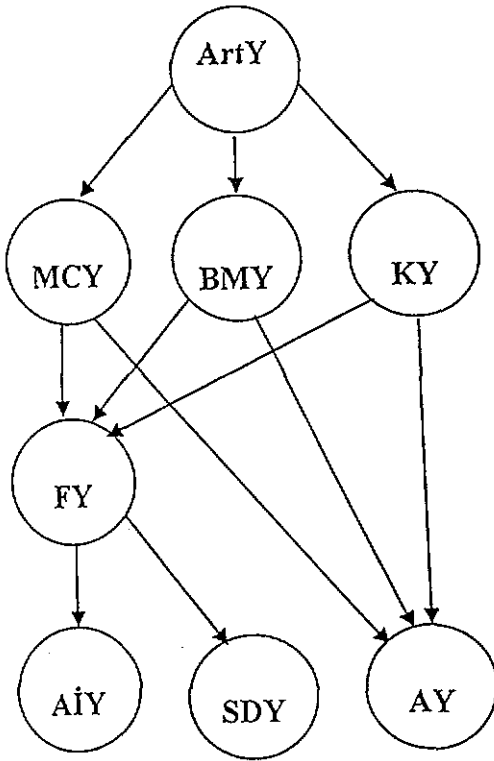
Tablo 2.1'de görüldüğü gibi, FY (farklar yöntemi) U_{rel} (görelî) tip fazlalığı gidererek U_{for} (biçimsel) tip fazlalık üretir. Bu ise SDY(sınıflandırma diyagramları yöntemi) veya AIY(ayrıştırıcı işaretler yöntemi) tarafından giderilir. Yani,

$$FY < SDY; \quad FY < AIY \quad (2.2)$$

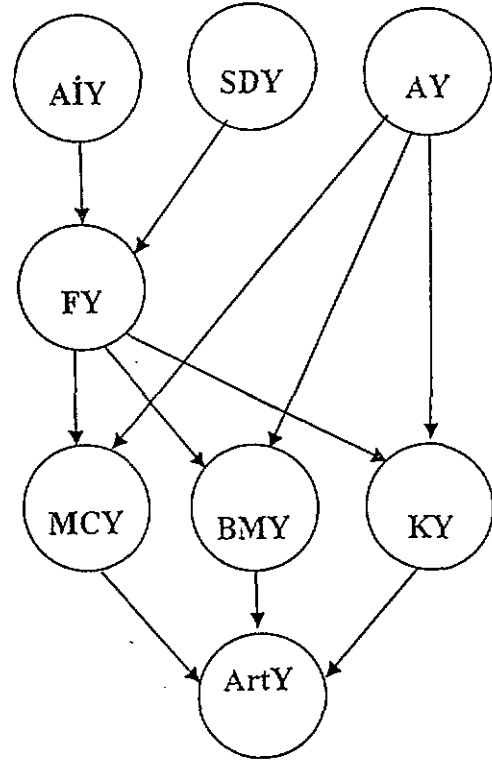
Aynı yolla aşağıdaki düzen de tespit edilir.

$$MCY < FY; \quad KY < FY; \quad BMY < FY; \quad MCY < AY; \quad KY < AY; \quad BMY < AY \quad (2.3)$$

(2.1)-(2.3) ifadeleri, sıkıştırma yöntemleri arasında, şekil 2.1 ve 2.2’de gösterilmiş olduğu gibi bir “Önceki-Sonraki” ilişkisi olduğunu göstermektedir.



Şekil 2.1. Sıkıştırma için sıra grafi



Şekil 2.1. Açma için sıra grafi

Şekil 2.1’de görüldüğü gibi sıkıştırma yöntemleri kümesine uygulanmış olan düzenleme ilişkisi aşağıdaki yollardan oluşan bir graf belirlemektedir:

Y1(ArtY, MCY, FY, AİY)

Y6(ArtY, KY, FY, AY)

Y2(ArtY, MCY, FY, SDY)

Y7(ArtY, MCY, AY)

Y3(ArtY, BMY, FY, AİY)

Y8(ArtY, BMY, SDY)

Y4(ArtY, BMY, FY, SDY)

Y9(ArtY, KY, SDY)

Y5(ArtY, KY, FY, AY)

Bu yollardan görüldüğü gibi:

1. Yollardan her biri sıkıştırma yöntemlerinin düzenlenmiş bir kümesini belirlemektedir
2. Bu kümelerden hiç biri, bütün fazlalık türlerini gideremezler. Çünkü, Y1-Y6 yolları FY'yi içerir ki bu da karakter tip bilgilere uygulanamaz. Y7-Y9 yolları ise U_{art} tip fazlalıkları giderecek yöntemler içermiyor. Dikkatli inceleme, bu yollarda diğer eksikliklerin de olduğunu göstermektedir.

Gösterilen yollardan elde edilen kümelere onların gidermediği fazlalıkları gideren yöntemleri eklemekle bunların her birini fonksiyonel tam duruma ulaştırmak mümkündür. Ancak bu iyi bir yol sayılamaz. Çünkü, bu yöntemlerden birileri diğerlerinin yerine geçebilir. Mesela, artışlar yöntemi, farklar yönteminin görevini yapabilir. Bu yüzden, AY olan yerde FY'ye gerek kalmayabilir. Bu yüzden, daha ciddi bir çözüm yolu bulmak gerekiyor.

2.2 Veri sıkıştırma yöntemlerinin fonksiyonel tam kümelerinin belirlenmesi

⁰
Böle fonksiyonları teorisinde F fonksiyonu ve onun implikantı f_i arasında aşağıdaki gibi bir ilişki vardır.

$$f_i = 1 \rightarrow F = 1$$

(2.4)

Böylece, eğer f_i termi F fonksiyonunu implikanti ise o zaman f_i termi F 'i 1'e eşit yapan mintermlerden en azı birini içermelidir. Buna benzer olarak ele bir R fonksiyonu seçelim ki, her hangi $M_i \in H_{\max}$ yöntemi her hangi fazlalığı giderecek durumda olduğunda $R=1$ olsun, yani

$$M_i: C_i \rightarrow \emptyset \rightarrow R=1 \quad (2.5)$$

(2.4) ve (2.5) ifadelerinin formal benzerliği ^{ndektir} sıkıştırma yöntemlerinin minimal kümesinin bulunması probleminin tek çıkışlı boole fonksiyonları için sade implikantların minimal kümesinin bulunması probleminin aynı olduğu görülebilir. Problemin çözümüne başlamadan önce fazlalık tiplerini sayısal ve karakter tip verilere uygulanıp uygulanamayacağına göre bir az daha detaylandırılalım. Hatırlatalım ki, şimdiye kadar ele alınmış olan fazlalık tipleri aşağıdakilerdir.

$$Q = \{U_{for}^s, U_{for}^k, U_{rel}^s, U_{rel}^k, U_{TD}^s, U_{TD}^k, U_{OE}^s, U_{OE}^k, U_{Art}^s\} \quad (2.6)$$

Yukarıda gösterildiği gibi, sıkıştırma yöntemlerinin maksimal kümesi ise aşağıdaki gibidir.

$$H_{\max} = \{ArtY, BMY, KY, MCY, FY, AY, ArtY, SDY\} \quad (2.7)$$

Sonraki işlemleri kolaylaştırmak amacı ile $H_{\max}$ kümesinin elemanlarını sırası ile M_1 - M_9 olarak gösterelim:

$$H_{\max} = \{M_1, M_2, M_3, M_4, M_5, M_6, M_7, M_8\} \quad (2.8)$$

$H_{\max}$ ve Q kümelerinin her bir elemanı, örtme tablosunun uygun olarak bir satırını ve sütununu göstermektedir (Tablo 2.2). “Sıkıştırma yöntemi M_i , U_j fazlalık tipini giderir” ilişkisinin (Gideren-Giderilen ilişkileri) olumluluk değeri uygun satır ve sütunun kesişme noktasında * olarak gösterilir.

Tablo 2.2. Sıkıştırma yöntemleri ile fazlalık tipleri arasındaki “Gideren-Giderilen” ilişkileri

Fazlal. Metod	U_{for}^s	U_{for}^k	U_{rel}^s	U_{rel}^k	U_{TD}^s	U_{TD}^k	U_{OE}^s	U_{OE}^k	U_{Art}^s
M_1									*
M_2							*	*	
M_3					*	*	*	*	
M_4					*	*			
M_5			*						
M_6			*	*					
M_7	*	*							
M_8	*	*							

Bu tablodan Petrik örtükleri aşağıdaki şekilde alınır

$$1. \forall (j = 1, 2, \dots, 9) (D_j = \bigcup_{i=1}^8 Z_{ji} \times M_i)$$

$$3. R = \bigcap_{j=1}^9 D_j$$

Bu ifadelerde $Z_{ji} \in \{0, 1\}$, j satır ile i sütununun kesişme noktasındaki değerdir.

Gösterilen çevirmeler Tablo 2.2’ye uygulandığında, aşağıdaki ifadeler elde edilir.

$$R = M_1 (M_2 + M_3)(M_2 + M_3)(M_3 + M_4)(M_3 + M_4)M_6 (M_5 + M_6)(M_7 + M_8)(M_7 + M_8)$$

Bu ifade, lojik kurallar uygulamakla aşağıdaki sade şekle gelir.

$$R = M_1 M_3 M_6 M_7 + M_1 M_3 M_6 M_8 + M_1 M_2 M_4 M_6 M_7 + M_1 M_2 M_4 M_6 M_8$$

Böylece, aşağıdaki minimum kümeler elde edilmiş olur.

$$\begin{aligned} H_1 &= \{M_1, M_3, M_6, M_7\}; & H_2 &= \{M_1, M_3, M_6, M_8\}; \\ H_3 &= \{M_1, M_2, M_4, M_6, M_7\} & H_4 &= \{M_1, M_2, M_4, M_6, M_8\} \end{aligned}$$

Bu kümelerden her biri minimal olup, (2.6)'da gösterilmiş olan bütün fazlalık terimlerini giderebilir. İşaretlerin yerine yöntemlerin isimleri konulursa, hem minimum hem de fonksiyonel tam olan aşağıdaki kümeler elde edilir:

$$\begin{aligned} H_1 &= \{ArtY, KY, AY, AİY\}; & H_2 &= \{ArtY, KY, AY, SDY\}; \\ H_3 &= \{ArtY, BMY, MCY, AY, AİY\}; & H_4 &= \{ArtY, BMY, MCY, AY, SDY\}; \end{aligned}$$

Görüldüğü gibi, kümelerin tümü ArtY ve AY'yi içermektedirler. H_1 ve H_2 kümeleri yalnız U_{for} fazlalığını gideren yöntemle farklılar. Aynı farklılık H_3 ve H_4 arasında da var. H_1 ve H_2 kümeleri ile H_3 ve H_4 kümeleri arasındaki fark ise, birincilerin kullandığı KY yerine ikincilerin (MCY, BMY) çiftini kullanmasından ibarettir. Böylece, bu kümelerin hepsi fonksiyonel olarak bir birine eşdeğerdir. Ancak H_1 ve H_2 daha az yöntem içerdiği için daha çok tercih edilmesi gereken durumdadırlar.

2.3. Veri sıkıştırma yöntemlerinin fonksiyonel tam kümelerinin sıkma başarısı açısından karşılaştırılması

Sıkıştırma yöntemlerinin minimum ve fonksiyonel tam olan kümesinin belirlenmesi, onun OKS' lerin program sistemlerinde yer alması açısından çok önemlidir. Yukarıda, böyle kümeler olarak aşağıdakiler elde edilmiş oldu.

$$H_1 = \{ArtY, KY, AY, AIY\}; \quad H_2 = \{ArtY, KY, AY, SDY\};$$

Şimdi ise bu kümelere giren yöntemlerle girmeyen yöntemleri etkililik açısından karşılaştırmaya çalışalım. Bu iş seçilmiş olan kümelerin gerçekten de en etkili olduğunu inanmak için gerekmektedir. Bunun için minimal kümelere girmeyen yöntemleri aşağıdaki gibi belirleyelim.

$$T = H_{\max} \setminus \{H_1, H_2\} = \{FY, BMY, MCY\}$$

Tabii ki, $M_i \in T$ ve $M_j \in \{H_1, H_2\}$ yöntemleri yalnız o halde karşılaştırılabilir ki, onların her ikisi de aynı bir fazlalık tipini gidersin. Bu açıdan, Tablo 2.1'den görülür ki, $FY \in T$ ancak $AY \in \{H_1, H_2\}$ ile ve $BMY, MCY \in T$ ise yalnız, $KY \in \{H_1, H_2\}$ ile karşılaştırılabilir. Böylece, yalnız

$$K_{AY} \geq K_{FY}; \quad (2.9)$$

$$K_{AIY} \geq K_{BMY} \quad (2.10)$$

$$K_{AIY} \geq K_{MCY} \quad (2.11)$$

durumlarda seçilmiş olan kümeler en iyi çözüm olarak değerlendirilebilirler. Bu yüzden, aşağıda bu değerlendirmeler yapılacaktır. Burada kullanılmış olan Kg, g isimli yöntemin sağladığı sıkma katsayısıdır.

1. $K_{AY} \geq K_{FY}$ şartının sağlandığı durumlar

Bu şartın kontrolü için $\dot{s}'_i = \pm(\dot{s}_e - \dot{s}_i)$ farkının sol tarafında meydana gelecek sıfırlar zincirinin uzunluğunun $\dot{s}_e$ ve $\dot{s}_i$ değerlerinin sol taraflarında olan aynı değerli basamaklar zincirinden uzun olmayacağını ispatlamak gerekir.

Öneri. Eğer $\dot{s}_e$ ve $\dot{s}_i$ değerleri soldan k. basamaktan sonra farklılarsa, o zaman onların farkının sol tarafında olan sıfırların sayısının en büyük olasılıklı değeri $r = k-1$ olacaktır.

İspatı. Farz edelim ki,

$$\dot{s}_e = \langle a_j^e \rangle_{j=1,2,\dots,n} \quad (2.12)$$

$$\dot{s}_i = \langle a_j^i \rangle_{j=1,2,\dots,n} \quad (2.13)$$

$$\dot{s}_e > \dot{s}_i \quad (2.14)$$

Bunların tümü, aşağıdaki gibi ifade edilebilir.

$$\exists(k \in \{1,2,\dots,n\})[\forall(j < k)[a_j^i = a_j^e] \& a_k^i < a_k^e] \quad (2.15)$$

Bu durumda, $\dot{s}'_i = (\dot{s}_e - \dot{s}_i)$ yalnız

$$a_k^e - a_k^i = 1 \quad (2.16)$$

olduğunda $n-r$ 'den az sayıda değerli (sıfır olmayan) rakama sahip olur. Bu yüzden (2.16) şartının ödendiği durumların olasılığını bulalım.

Eğer s_i ve s_e değerleri Q tabanlı sayı sisteminde verilmişse, o zaman a_k^e basamağının (2.16) şartını ödeyen $Q-1$ değişik değeri olabilir. a_k^e 'nin her bir değerine a_k^i 'nin $a_k^i \in \{0,1,\dots,a_k^e-1\}$ değerlerinden bir uygun gelebilir. Bu durumda (2.15) şartını ödeyen $\langle a_k^e, a_k^i \rangle$ çiftlerinin sayısı aşağıdaki gibi belirlenebilir.

$$P = \frac{Q(Q-1)}{2} \quad (2.17)$$

Bunlardan yalnız $Q-1$ sayıdaki çift (2.16) şartını öder. Bu durumların meydana gelmesi olasılığı ise aşağıdaki gibi belirlenebilir.

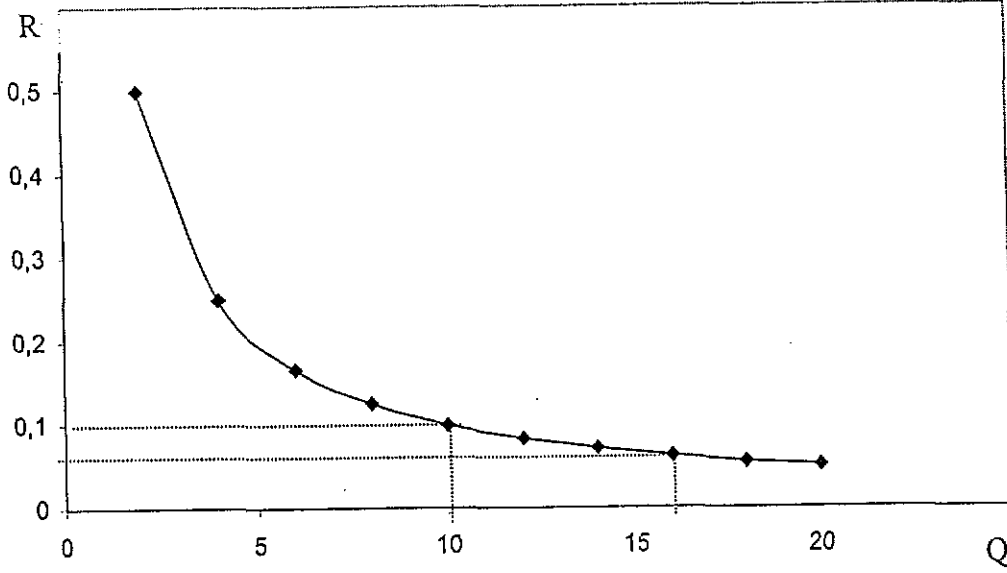
$$P_1 = \frac{Q-1}{P} = \frac{2}{Q} \quad (2.18)$$

a_k^e 'den borç alınma olasılığını $P_2=0.5$ olarak kabul edelim. O zaman, $k-1$ sayıda sol basamağın aynı olması durumunda $s_i' = s_e - s_i$ farkının değerli basamaklarının sayısının $n-r$ 'den az olması olasılığı aşağıdaki gibi belirlenebilir.

$$R = P_1 P_2 = \frac{2}{Q} \times 0.5 = Q^{-1} \quad (2.19)$$

Şekil 2.3'de Q 'nün pratik değerleri için $R = Q^{-1}$ bağıllığının eğrisi gösterilmiştir.

Bu şekilden de görüldüğü gibi, $Q \geq 10$ durumu için $R \leq 0.1$ olduğu için artışlar yöntemi, farklar ve ayrıştırıcı işaretler yönteminin birlikte sağladığı kadar sıkma katsayısı sağlayacaktır.



Şekil 2.3. $R=1/Q$ bağılılığı

2. (2.10) şartının sağlandığı durumlar

Bu şartın sağlandığı durumları tespit edebilmek için aşağıdakileri inceleyelim:

m : birlikte sıkıştırılacak serilerin sayısı; L_i : $\bar{S}_i$ serisinin uzunluğu; M : $\hat{S}_0 = \bigcup_{i=1}^m \bar{S}_i$

kümesinin büyüklüğü; $l(c)$: bir sembolün bit cinsinden uzunluğu; $l(f)$: $\hat{S}_0$ kümesinin bir elemanı için ayrılmış olan alanın sembol cinsinden uzunluğu.

$\bar{S}_1, \bar{S}_2, \dots, \bar{S}_m$ serilerinin sıkılmamış durumları için gereken bellek miktarı (bit cinsinden) aşağıdaki gibi belirlenebilir.

$$W(\{\bar{S}_i\}_{i=1,2,\dots,m}) = l(c) \times l(f) \times \sum_{i=1}^m L_i \quad (2.19)$$

Bu seriler KY veya BMY ile sıkıştırıldığında, esas bilginin miktarı aşağıdaki gibi belirlenecektir.

$$W(\hat{S}_0) = M \times l(f) \times l(c) \quad (2.20)$$

KY ve BMY tarafından üretilmiş olan yardımcı bilginin miktarı ise aşağıdaki gibi belirlenir.

$$W(D_K) = \lceil \log_2 M \rceil \times \sum_{i=1}^m L_i \quad (2.21)$$

$$W(D_{BM}) = m \times M \quad (2.22)$$

KY ve BMY'nin daha etkili olması için aşağıdaki şartların ödenmesi gerekir:

$$W(\hat{S}_0) + W(D_K) < W(\{\bar{S}_i\}_{i=1,2,\dots,m}) \quad (2.23)$$

$$W(\hat{S}_0) + W(D_{BM}) < W(\{\bar{S}_i\}_{i=1,2,\dots,m}) \quad (2.24)$$

(2.19)-(2.22) dikkate alınmakla (2.23) ve 2.24)'den aşağıdakiler elde eldir.

$$l(c) \times l(f) \times M + \lceil \log_2 M \rceil \times \sum_{i=1}^m L_i < l(c) \times l(f) \times \sum_{i=1}^m L_i \quad (2.25)$$

$$l(c) \times l(f) \times M + m \times M < l(c) \times l(f) \times \sum_{i=1}^m L_i \quad (2.26)$$

(2.25) ve (2.26) ifadelerini kullanarak KY ve BMY yöntemlerinin sağladığı

sıkıştırma katsayıları bulunabilir:

$$K_{KY} = \frac{l(c) \times (lf) \times \sum_{i=1}^m L_i}{M \times l(c) \times l(f) + \lceil \log_2 M \rceil \times \sum_{i=1}^m L_i} \quad (2.27)$$

$$K_{BMY} = \frac{l(c) \times (lf) \times \sum_{i=1}^m L_i}{M \times l(c) \times l(f) + m \times M} \quad (2.28)$$

(2.27) ve (2.28)'den görüldüğü gibi, (2.10) şartı aşağıdaki durumda ödenir.

$$\lceil \log_2 M \rceil \times \sum_{i=1}^m L_i \leq m \times M \quad (2.29)$$

Serilerin toplam uzunluğu aşağıdaki gibi hesaplanabilir.

$$\sum_{i=1}^m L_i = m \times L_{orta} \quad (2.30)$$

Bunu dikkate alarak, $K_{KY} \geq K_{BMY}$ olması için aşağıdaki şartın yerine getirilmesi gerekir.

$$L_{orta} \leq \frac{M}{\lceil \log_2 M \rceil} \quad (2.31)$$

Tablo 2.3'de $\lceil \log_2 M \rceil$, L_{orta} ve $L_{orta} = 100 \times \frac{L_{orta}}{M}$ değerleri verilmiştir.

Tablo 2.3. Bir serinin orta uzunluğunun serilerin elemanları kümesinin büyüklüğüne bağlılığı

M	$\lceil \log_2 M \rceil$	L_{orta}	$L_{orta} = 100 \times \frac{L_{orta}}{M}$
2	1	2	100
4	2	2	50
8	3	3.6	33.3
16	4	4	25
32	5	6.4	20
64	6	10.7	15
128	7	18.3	13
256	8	32	12.5
512	9	59	11.1
1024	10	102,4	10
2048	11	186	9
4096	12	341	8.5
8102	13	630	7.7
16284	14	1161.7	7.1
32568	15	2171	6.1
65136	16	4071	6.25

Gerçek serilerin uzunlukları, Tablo2.3'te gösterilmiş olan maksimum uzunluğu aşmıyor [41,82]. Bu durumda $M / \lceil \log_2 M \rceil = 65136 / 16 = 4096$. Buna uygun gelen

$L_{orta} = 6.25 \ll 4096$ olduğu için

$$L_{opta} \leq \frac{M}{\lceil \log_2 M \rceil}$$

şartı ödenir ki, bu da pratik olarak her zaman $K_{KY} \geq K_{BMY}$ olacağını göstermektedir.

3. (2.11)'in ödenmesi şartlarının karşılandığı durumlar

KY ve MCY analizinden belli olduğu gibi, U_{TD} tip fazlalığın giderilmesi açısından KY ve MCY aynı derecede etkilidirler. Çünkü, her ikisinin de ürettiği esas veri aynıdır. KY'nin ürettiği yardımcı bilginin miktarı (2.21) formülü ile belirlenir. Yani,

$$W(D_{KY}) = L \times \lceil \log_2 M \rceil \quad (2.32)$$

MCY-m yönteminin ürettiği yardımcı bilginin miktarı ise aşağıdaki formülle hesaplanabilir. $\bar{S} = \{s_i\}_{i=1,2,\dots,L}$ serisinde M değişik değer ve bu değerlerin tekrarlanma sayısı $R_1, R_2, \dots, R_M$ olsun. Bu durumda MCY' nin birinci uygulamasından sonra aşağıdaki lojik cetvel elde edilmiş olur.

$$Y_1 = \sum_{i=1}^M R_i \quad (2.33)$$

Bu yöntemin m defa uygulanmasından sonra uzunluğu aşağıdaki gibi belirlenen bir lojik cetvel elde edilir.

$$Y_m = \sum_{i=m}^M R_i \quad (2.34)$$

Tabii ki, MCY'nin art arda uygulamalarının sayısı N-1'i aşamaz. Buna göre de en son alınan mantık cetvelinin uzunluğu aşağıdaki gibi olacaktır.

$$Y_{M-1} = \sum_{i=M-1}^M R_i \quad (2.35)$$

Buna göre de MCY-m tarafından üretilmiş olan mantık cetvellerinin toplam uzunluğu aşağıdaki gibi belirlenecektir.

$$W(D_{MC}) = \sum_{j=1}^{M-1} Y_j \quad (2.36)$$

(2.34) ve (2.35)'i (2.36)' da dikkate alarak aşağıdaki ifade elde edilebilir.

$$W(D_{MC}) = \sum_{i=1}^M i \times R_i - R_M$$

Genellikle, $R_M \ll \sum_{i=1}^M i \times R_i$ olduğunu için

$$W(D_{MC}) = \sum_{i=1}^M i \times R_i \quad (2.37)$$

$P_i = R_i/L$ olduğu dikkate alınırsa

$$W(D_{MC}) = L \times \sum_{i=1}^M i \times P_i \quad (2.38)$$

Burada $\sum_{i=1}^M i \times P_i$ ifadesi sembol başına düşen bit sayısının orta değerini gösterdiği

için, bu sıkıştırma yöntemi, değişik uzunluklu optimal kodlama (DUOK) yönteminin bir şekil değişmesi olarak meydana çıkmaktadır. Benzer olarak, KY' de $R_i = R_j$ durumunda DUOK yönteminin diğer bir şekli olarak meydana çıkmaktadır. Böylece, KY ve MCY-m yöntemleri, aşağıdaki durumlarda maksimum sıkıştırma katsayısı sağlayabilirler.

$$\forall(i, j \in \{1, 2, \dots, M\})(R_i = R_j)$$

$$\forall(i, j \in \{1, 2, \dots, M-1\})(R_{i+1} = 0.5 \times R_i)$$

Bu şartlardan hiç biri ödenmediğinde, KY ve MCY-m'in görelî etkililiği aşağıdaki şarttan belirlenebilir.

$$\lceil \log_2 M \rceil \leq \sum_{i=1}^M i \times P_i \quad (2.39)$$

MCY-M'de DUOK yöntemi gibi değişik uzunluklu mantık cetvellerinden oluşan kod açma tablosu kullanılmaktadır. Bu ise kullanılan alfabenin sembolleri sayısı çok oldukça açma için daha çok zorluk çıkarmaktadır [19]. Bu yüzden, söz konusu yöntem, yalnız alfabede bulunan sembollerin sayısı 10'un birkaç katsayısını aşmadığı hallerde kullanılabilir. [59]'da MCY-m ikili serilere uygulanmış ve bir sembol olarak 3 bit kabul edilmiştir. Bu yöntemin etkililiği sembollerin dağılım frekansına çok bağlı olduğu için, yaklaşık olarak sabit kalan arşiv verilere uygulanması daha doğru olabilir.

Genellikle, OKS'lerde kullanılan kayıtların uzunluğu 30-300 bayt arasında, tipik olarak ise 100 bayt civarında bulunmaktadır [3]. Bu durumda, KY her bir sıkıştırılmış kayıt için 1-2 baytlık yardımcı bilgi kullanmaktadır ki, bu da kaydın uzunluğunun çok cüzi bir kısmıdır. Bundan başka, KY kayıtlara direkt başvuru imkanı sağlar ve hata trek'ine sebep olmaz. Bu yüzden, sıkıştırma katsayısı açısından DUKY ve MCY-m karşısında biraz kayıp etmesine rağmen, KY'nin minimum fonksiyonel tam kümeye dahil edilmesi doğru bir seçim olarak değerlendirilebilir.

2.4. Kortejleri iki taraflı kesilme yöntemi

Tablo 2.1’de görüldüğü gibi, yalnız KY aynı zamanda iki fazlalık tipini (U_{TD} ve U_{OE}) giderebilir. Paragraf 2.1’de gösterilmiş olduğu gibi, ArtY’den sonra H_1 ve H_2 kümelerinden olan her hangi bir yöntem uygulanabilir. Aynı durum KY için de geçerlidir. Çünkü o da sonuç olarak tekrarlanmayan değerlerin düzenli bir serisini üretir. Bu durumda, U_{for} veya U_{rel} tip fazlalıklar bulunduran verilere KY uygulanırsa, o fazlalıklar bir az azalmış olsalar bile yine de sonuçta kalırlar. Lakin AY, AİY ve SDY Farklı şekilde çalışırlar. AY’nin uygulanması U_{rel} tip fazlalığı, AİY ve SDY ise U_{for} tip ^{fazlalığı} giderir. Aynı zamanda hem U_{for} hem de U_{rel} tip fazlalık bulunduran verilere önce FY sonra ise AİY veya SDY uygulamak gerekir. Fakat 2.2’de gösterilmiş olduğu gibi, FY karakter tip bilgileri için yaramıyor. AY ise U_{rel} tip fazlalığı gidermekle $S = \langle \langle \dot{s}_i, * \rangle \rangle_{i=1,2,\dots,N}$ veri yapısı oluşturur. Bu yapıya ise ne AİY nede SDY uygulanamaz. Eğer U_{for} tip fazlalık sol doldurucu işaretler zincirlerinden dolayı meydana çıkıyorsa, o zaman bu fazlalık U_{rel} fazlalığı ile birlikte tam ve ya kısmi olarak giderilebilir. Fakat söz konusu fazlalık sağ doldurucu semboller zincirlerinden dolayı meydana çıkıyorsa, bunu her hangi bir yöntemle gidermek mümkün olmuyor.

Bölüm 2.2’de sol zincirlerin var olmasından meydana çıkan U_{for} fazlalığı, U_{for}^s ve U_{for}^k olarak ikiye bölünmüştü. Bunlardan birincisi sayısal verilere, ikincisi ise karakter tip verilere uygulandığı için ikisi bir veride olamaz. Fakat U_{for} ve U_{rel} fazlalık tipleri ise yoğun olarak rastlanmaktadır. Mesela, sanayi müesseselerinin pasaportlarında [73], “Belgenin ismi”, “Özellğin adı”, “Deskriptorlar”, “Meslek”, “Ürünün tipi”, “Teknoloji”, “Fonksiyonel görevi” vs. nitelikler değişik uzunluklu değerler alabilirler, ancak, bunlar için ayrılmış olan alanın uzunluğu en uzun değere göre belirlenir [5]. Bu tür nitelikler, genellikle anahtar olarak kullanılır ve bu yüzden onların değerleri düzenlenmiş

olur. Bu ise U_{for} ve U_{rel} tip fazlalıkların birlikte meydana çıkmasına sebep olmaktadır. Böylece, U_{for} ve U_{rel} tip fazlalıkları aynı zamanda giderebilecek bir sıkıştırma yöntemine ihtiyaç olduğu görülmektedir.

Farz edelim ki, her hangi bir niteliğin değerleri $\bar{S} = \{\bar{s}_i\}_{i=1,2,\dots,N}$ serisi olarak verilmiştir. Her bir i için $\bar{s}_i \leq \bar{s}_{i+1}$ ve bir $\bar{s}_{ie}$ olduğunu var sayalım. n -küpler üzerinde koordinatlı çıkarma işlemine [81] benzer olarak, kortejler üzerinde çıkarma işlemi yapalım. Bunun için $\bar{s}_i = \langle a_j^i \rangle_{j=1,2,\dots,n}$, $\bar{s}_{ie} = \langle a_j^{ie} \rangle_{j=1,2,\dots,n}$ olduğunu kabul edelim. Bu durumda, fark korteji aşağıdaki gibi belirlenebilir.

$$\bar{s}_{ir} = \bar{s}_i \# \bar{s}_{ie} = \langle z_j^i \rangle_{j=1,2,\dots,n} \quad (2.40)$$

Burada

$$z_j^i = \begin{cases} - \text{ eğer } a_j^i = a_j^{ie} \\ a_j^i \text{ eğer } a_j^i \neq a_j^{ie} \end{cases} \quad (2.41)$$

(2.41)'e paralel olarak x_j^i basamağı aşağıdaki gibi belirlenen bir mantık cetveli de oluşturulur.

$$x_j^i = \begin{cases} 0, \text{ eğer } a_j^i = a_j^{ie} \\ 1, \text{ eğer } a_j^i \neq a_j^{ie} \end{cases} \quad (2.42)$$

Böylece, (2.41) ve (2.42) dönüştürmelerinden sonra $\bar{s}_i$ korteji $\langle \bar{s}'_i, y_i \rangle$ çiftine çevrilir. Örnek olarak bölgesel OKS'in veri tabanında bulunan "Memulun adı" niteliğinin komşu olan 2 kaydını ele alalım [30,31,73].

$$\bar{s}_i = \langle TBPV_127_CT.40XH \rangle$$

$$\bar{s}_{iv} = \langle TBPV_114_CT.45_ \rangle$$

$$\bar{s}_{ir} = \langle \text{-----}27\text{----}0XH \rangle$$

$$y_i = \langle 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 1\ 1\ 1 \rangle$$

Yani,

$$\bar{s}'_i = \forall (a'_j)(z'_j \neq -) \quad (2.43)$$

Görüldüğü gibi, y_i mantık cetveli varken boş sembolleri tutmaya gerek yoktur. Bu yüzden yalnız $\bar{s}'_i = \langle 270XH \rangle$ kortejini saklamak yeterlidir. Bu yolla sıkılmış olan veri aşağıdaki şekilde açılır.

$$a'_j = \begin{cases} s'^{iv}_j, & \text{eğer } x^i_j = 0 \\ s'^{ir}_{kj}, & \text{eğer } x^i_j = 1 \end{cases}$$

$$\text{Burada } kj = \sum_{l=1} x^l_i, \quad j=1,2,\dots,n,$$

(2.41) ve (2.43)'de görüldüğü gibi, $z_j^i = -\longleftrightarrow a_j^i = a_j^{ie}$. Yani (2.43) şartı aşağıdaki gibi gösterilebilir.

$$\bar{s}_i' = \forall(a_j^i)(a_j^i \neq a_j^{ie}) \quad (2.44)$$

Bu ifadeden görüldüğü gibi, $\bar{s}_i'$ direkt olarak $\bar{s}_i$ 'den elde edilebilir. (2.42) ve (2.44) dönüşümlerini $\bar{S}$ serisinin tümüne uygulamakla aşağıdaki veri yapısı elde edilebilir.,

$$S_y = \langle\langle \bar{s}_i', Y_i \rangle\rangle_{i=1,2,\dots,N} \quad (2.45)$$

Ancak (2.45) açma için elverişli değil. Çünkü değişik tabiatlı olan veriler (karakterler ve mantık cetvelleri) içermektedir. Bundan başka, kesilmiş kortejlerle mantık cetvellerinin birlikte tutulması veri gizliliğini imkansız kılmaktadır. Bu yüzden, (2.45)'i aşağıdaki şekle getirmek daha uygundur.

$$S_y = \{ \langle \bar{s}_i' \rangle_{i=1,2,\dots,N}, \langle Y_i \rangle_{i=1,2,\dots,N} \} = \{ \bar{S}', \bar{D} \} \quad (2.46)$$

Bu veri yapısını üreten sıkıştırma yöntemini Bileşensel Fark yöntemi (BFY) olarak tanımlayalım. Bu vesile ile (2.40) da gösterilmiş olan $\bar{D}$ kortejini yeniden ele alalım.

BFY, aynı zamanda U_{rel} ve U_{for} tip fazlalıkları giderdiği için, her bir mantık cetveli soldan r_i uzunluklu, sağdan da e_i uzunluklu 0'lar zincirine sahip olacaktır. Bu durumda , ortada $k_i = (n - r_i - e_i)$ uzunluklu bir kısım kalır. Bu şartlar dahilinde kesilmiş olan $\bar{s}_i$ kortejinin kesilmiş olan bileşenlerinin miktarı aşağıdaki gibi belirlenecektir.

$$W(s_{ik}) = W(g)(r_i + pk_i + e_i) \quad (2.47)$$

Burada, $W(g)$ bir bileşenin bit cinsinden uzunluğu, p ise mantık cetvelinin $(r_i + 1) \leq j \leq (r_i + k_i)$ numaralı bitinin 0 olması olasılığıdır.

$\forall((r_i + 1) \leq j \leq (r_i + k_i))(x_j^1 = 1)$ durumları için Y_i mantık cetvelini aşağıdaki şekilde tertiplemek daha uygundur.

$$Y_i = \langle r_i, k_i \rangle \quad (2.48)$$

Çünkü en kötü durumda bile oluşacak $\max r_i$ veya $\max k_i$ değerleri için yalnızca $\lceil \log_2 n \rceil$ bit gerekecektir. Başka bir deyişle,

$$\forall(n > 6)(W(\langle r_i, k_i \rangle) = 2 \times \lceil \log_2 n \rceil < n) \quad (2.49)$$

Bu ise Tablo 2.4'de açık bir şekilde görülmektedir.

Tablo 2.4. n ile $2 \times \lceil \log_2 n \rceil$ arasındaki bağıllık.

n	2	3-4	5-8	9-16	17-32	33-64	65-128	129-256
$2 \times \lceil \log_2 n \rceil$	2	4	6	8	10	12	14	16

Görüldüğü gibi, n 'in yükselmesi ile

$$K_L = \frac{n}{2 \times \lceil \log_2 n \rceil} \quad (2.50)$$

sıkıştırma katsayısı da hızla yükselmektedir. Lakin bu genel katsayıya zayıf bir şekilde yansır. Çünkü, her zaman

$$\forall((r_i + 1) \leq j \leq (r_i + k_i))(x_j^i = 1) \quad (2.51)$$

şartı ödenmez. Yi mantık cetvelinin (2.48) biçiminde saklanabilmesi için $\bar{s}_i$ kortejini aşağıdaki gibi kesmek gerekir.

$$\bar{s}_i' = \text{Pr}(\alpha_i, \beta_i) \bar{s}_i \quad (2.52)$$

Burada α_i, β_i aşağıdaki gibi belirlenir.

$$\forall(j < \alpha_i)((a_j^i = a_j^{ie}) \& (a_{ai}^i \neq a_{ai}^{ie})) \quad (2.53)$$

$$\forall(j < \beta_i)((a_j^i = a_j^{ie}) \& (a_{\beta i}^i \neq a_{\beta i}^{ie})) \quad (2.54)$$

Bu kurallara göre, kortejler sol ve sağ sınırlardan başlayarak karşılaştırılırlar. Mesela,

$$\begin{aligned} \bar{s}_i &= \langle TBPV_127_CT.40XH \rangle \\ \bar{s}_{ie} &= \langle TBPV_114_CT.45_ \rangle \end{aligned}$$

Bu kortejler (2.53) ve (2.54) kurallarına göre kesildiğinde aşağıdaki kesilmiş kortej ortaya çıkar.

Burada $\alpha_i = 7$; $\beta_i = 16$ olduğu için

$$\bar{s}_i' = \text{Pr}(\alpha_i, \beta_i) \bar{s}_i = \langle 27_CT.40XH \rangle$$

U_{rel} tip fazlalık her zaman sol tarafta bulunduğu için (2.54) şartını aşağıdaki gibi yazmak daha uygun olabilir.

$$\forall (j < \beta_i)((a_j^i = DS) \& (a_{\beta_i} \neq DS)) \quad (2.55)$$

Görüldüğü gibi, (2.52), (2.53) ve (2.55) kuralları kullanılarak yerine getirilmiş olan bir kesme, aynı zamanda U_{rel} ve U_{for} fazlalıklarının giderilmesine sebep olur. (2.55)'e göre elde edilmiş olan korteji açmak için gereken Y_i mantık cetveli ise aşağıdaki gibi belirlenecektir.

$$\forall (j < \alpha_i)(Pr_j Y_i = 0) \& \forall (\alpha_i \leq j \leq \beta_i)(Pr_j Y_i = 1) \quad (2.56)$$

Bu ise (2.48) şeklinde ifade edilebilir. Burada, $r_i = \alpha_i - 1$, $k_i = \beta_i + 1$.

Böylece, söz konusu yöntem, kesilmekte olan kortejin orta kısmına dokunmuyor. Bu yolla alınan veri yapısının genel ifadesi aşağıdaki gibi olur.

$$S_{DY} = \{\bar{S}'_{DY}, \bar{D}_{DY}\} \quad (2.57)$$

Tabii ki,

$$W(S_{DY}) \leq W(S_Y) \leftrightarrow W(\bar{S}'_{DY}) + W(\bar{D}_{DY}) \leq W(\bar{S}'_Y) + W(\bar{D}_Y) \quad (2.58)$$

(2.58)'in sağ tarafı aşağıdaki gibi yazılabilir.

$$W(\bar{S}'_{DY}) - W(\bar{S}_Y) \leq W(\bar{D}_Y) - W(\bar{D}_{DY}) \quad (2.59)$$

$$W(\bar{S}'_{DY}) - W(\bar{S}_Y) = P \times W(g) \times \sum_{i=1}^N K_i \quad (2.60)$$

$$W(\bar{D}_Y) - W(\bar{D}_{DY}) = N \times (n - 2 \times \lceil \log_2 n \rceil) \quad (2.61)$$

$$\bar{D}_Y = \langle \langle x_j^i \rangle_{j=1,2,\dots,n} \rangle_{i=1,2,\dots,N}, \quad \bar{D}_{DY} = \langle \langle r_i, k_i \rangle \rangle_{i=1,2,\dots,N} \text{ ve } \max l(r_i) = \max l(k_i) = \lceil \log_2 n \rceil$$

olduğu için ve (2.60), (2.61) ifadeleri dikkate alınırsa (2.59) aşağıdaki duruma gelir.

$$P \times W(g) \times \sum_{i=1}^N K_i \leq N \times (n - 2 \times \lceil \log_2 n \rceil) \quad (2.62)$$

Ki'nin tesadüfî karakterinden dolayı (2.62)'nin kesin değerinin hesaplanması imkansızdır. Bu işi kolaylaştırmak için aşağıdaki olguları dikkate alalım.

$$\sum_{i=1}^N K_i = N \times K_{orta}; \quad P = \frac{1}{Q}$$

Burada Q verileri temsil etmek için kullanılan alfabedeki harflerin sayısı, K_{orta} ise bir kesilmiş kortejin ortalama uzunluğudur. Bu durumda, (2.62) şartı aşağıdaki şekli alır.

$$K_{orta} \leq \frac{Q}{W(g)} \times (n - 2 \times \lceil \log_2 n \rceil) \quad (2.63)$$

Genellikle, Q=256 olduğu için

$$K_{orta} \leq 8 \times (n - 2 \times \lceil \log_2 n \rceil) \quad (2.64)$$

(2.64) şartının $n \geq 7$ durumunda yerine gelmesi, bütün pratik durumlarda S_{DY} veri yapısının S_Y yapısından daha sıkışık olacağı anlamına gelmektedir. S_{DY} veri yapısını üreten sıkıştırma yöntemi iki taraflı kesme yöntemi olarak tanımlanacaktır. S_{DY} ve S_Y yapılarının kortajlara direkt başvuru açısından araştırılması S_{DY} yapısının daha avantajlı olduğunu göstermiştir.

2. Bölümün Sonuçları

1. Birinci bölümde açıklanmış olan sıkıştırma yöntemlerinin giderilen ve üretilen fazlalık tipleri açısından incelenmesi aşağıdakileri göstermiştir:
 - Aynı tip fazlalığı gideren yöntemler düzen işlemine tabi tutulamazlar
 - Değişik tip fazlalıkları gideren yöntemler arasında bir düzen vardır
 - En çok sayıda fazlalık tiplerini giderebilen yöntemler kümeleri mevcuttur
2. Boole fonksiyonları için minimum örtme yaklaşımına dayanarak, sıkıştırma yöntemlerinin minimum kümeleri belirlenmiş, bunların fonksiyonel tam oldukları ispatlanmıştır. Bu kümelerde olan yöntemler performans açısından değerlendirilmiş, özellikle de bir birine alternatif gibi görünen Artışlar ve kodlama yöntemleri ile Farklar ve Bit matrisleri ayrıntılı olarak incelenmiş ve birincilerin daha yüksek sıkıştırma katsayısı sağlayabileceği gösterilmiştir.
4. Dosyalarda bulunan fazlalık tiplerinin araştırılması göstermiştir ki, bazı dosyalarda U_{rel} ve U_{for} tip fazlalıklar birlikte bulunurlar. Bunlardan birinin giderilmesi için uygulanan bir yöntem, diğerinin giderilmesini sağlayacak başka bir yöntemin uygulanmasını imkansız kılmaktadır.
5. Bu olumsuzluktan kurtulabilmek için söz konusu fazlalıkların ikisini de birden gideren bir yöntem geliştirilmiştir.
6. Sıkıştırma amacı ile kullanılan veri işleme prosedürleri analiz edilmiş, bunların her birinin kullandığı ve ürettiği veri yapıları belirlenmiş ve bu yolla sıkıştırma derleyicisi adı verilmiş olan programa karşı koyulması gereken talepler belirlenmiştir.

3. VERİ SIKIŞTIRMA PROSESLERİNİN ALGORİTMALAŞTIRILMASI, BU ALGORİTMALARIN BELİRLİ PROSEDÜRLERE AYRILMASI VE SIKMA VE AÇMA İÇİN BİRLEŞMİŞ ALGORİTMALARIN TERTİPLANMESİ

3.1 . Ayrıştırıcı işaretler (AİY) ve sınıflandırıcı diyagramlar (SDY) yöntemlerinin algoritmaları

Kortejler AİY ve SDY' ye göre aynı şekilde kesildiğine göre bu yöntemlerin algoritmaları da ortak bir prosedüre sahip olmalıdırlar. Bu prosedür G_1^1 olarak Tablo 3.1' de gösterilmiştir.

Tablo 3.1. G_1^1 prosedürü

Etiket	İşlem	Açıklama
Y1	$\bar{s}_i$ nin okunması	$i=1,2,...,N$
Y2	$j=1$	Birinci bileşenin indisi
Y3	$r_i=0$	
Y4	Eğer $a_j^i = DS$ ise Y5,	geçerli bileşenin dolgu sembolü (DS) ile karşılaştırılması
Y5	$r_i = r_i + 1$	DS zincirinin uzunluğunun hesabı
Y6	$j=j+1$	bir sonraki bileşenin indisi
Y7	Y4'e atlama	Bir sonraki bileşenin karşılaştırılmasına başlamak
Y8	$k_i = n - r_i$	Kesilmiş kortejin uzunluğu
Y9	$\bar{s}_i' = \text{Pr}(r_i + 1, n)\bar{s}$	Kesilmiş kortejin seçilip ayrılması
Y10	$[A_s] = \bar{s}_i'$	Kesilmiş kortejin A_s adresine yazılması
Y11	$A_s = A_s + k_i$	Ayrıştırıcı işaretin adresi
Y12	SON	G_1^1 prosedürünün sonu

G_1 prosedürüne göre, kesilmiş kortejler direkt olarak esas belleğe yazılırlar. İleride, farklılaştırma amacı ile, Sıkıştırılmaya ait olan isimlere -C, açmaya ait olan isimlere ise -P eklenecektir.

AIY-C ve SDY-C'ye göre G_1' prosedürünü, sıkıştırılmış veri yapısının tertiplenmesi prosedürü izlemektedir. Bu yöntemler farklı veri yapıları ürettikleri için bunlar için gereken $G_{1,2}$ ve $G_{2,2}$ prosedürleri de farklıdır. AIY-C ve SDY-C uygun olarak aşağıdaki veri yapılarının üretirler:

$$S_P = \langle \langle \bar{s}_i', * \rangle \rangle_{i=1,2,\dots,N} \quad (3.1)$$

$$S_K = \{ \langle \bar{s}_i', * \rangle_{i=1,2,\dots,N}, \langle K_i \rangle_{i=1,2,\dots,N} \} \quad (3.2)$$

Tablo 3.2'de $G_{1,2}$ prosedürünün açıklanması verilmiştir. Bu prosedür * işareti ekleme işlemini yerine getirmektedir.

Tablo 3.2. $G_{1,2}$ prosedürü

Etiket	İşlem	Açıklama
Q1	$[A_s]=*$	Ayrıştırıcı işaretin eklenmesi
Q2	$A_s = A_s + 1$	Bir sonraki kesilmiş kortejin yazılacağı adres
Q3	SON	$G_{1,2}$ prosedürünün sonu

G_1' ve $G_{1,2}$ prosedürleri bir $\langle \bar{s}_i', * \rangle$ çifti oluşturur. Fakat (3.1) veri yapısını oluşturmak için $G_{1,3}$ prosedürü olarak i'ye göre döngü oluşturmak gerekir (Tablo 3.3).

Tablo 3.2. $G_{1,3}$ prosedürü

Etiket	İşlem	Açıklama
T1	$i = i+1$	Bir sonraki kortejin indisi
T2	Eğer $i \leq N$ ise Y1	G_1^1 'e şartlı atlama
T3	SON	$G_{1,3}$ prosedürünün sonu

Tabii ki, G_1^1 'den önce parametrelere başlangıç değerler aktaran bir G_0 prosedürünün olması gerekir (Tablo 3.4).

Tablo 3.4. G_0 prosedürü

Etiket	İşlem	Açıklama
I1	$i=1$	Birinci kortejin indisi
I2	$N=<\text{değer}>$	Kortejlerin sayısı
I3	$A_s=<\text{değer}>$	Kesilmiş kortejler serisinin başlangıç adresi
I4	$n=<\text{değer}>$	Geçerli kortejin sonuncu bileşeninin indisi
I5	SON	G_0 prosedürünün sonu

Her zaman birlikte olacakları için G_0 ve G_1^1 kortejleri birlikte $G_{0,1}$ olarak gösterilirse, AIY -C 'nin algoritmik şeması aşağıdaki gibi gösterilebilir.

$$\overline{AIY-C} = \langle G_{0,1}, G_{1,2}, G_{1,3} \rangle \quad (3.3)$$

Şimdi de SDY -C 'nin oluşturduğu veri yapısını ele alalım.

$$S_K = \{\bar{S}_K', \bar{D}_K\}$$

Burada, $\bar{S}'_K = \langle \bar{s}'_i \rangle_{i=1,2,\dots,N}$, $\bar{D}_K = \langle K_i \rangle_{i=1,2,\dots,N}$

$\bar{S}'_K = \langle \bar{s}'_i \rangle_{i=1,2,\dots,N}$ alt yapısı G_1^1 prosedürünün Y8-Y11 işlemleri ile gerçekleştirilir.

$\bar{D}_K = \langle K_i \rangle_{i=1,2,\dots,N}$ alt yapısı ise aşağıdaki gibi elde edilir.

$\bar{D}_K = \langle K_i \rangle_{i=1,2,\dots,N}$ serisinin her bir elemanının uzunluğu $l(k_i) = \lceil \log_2 n \rceil$ olarak belirlenir. Bu yüzden, bu elemanların her biri için bir bayt ayırmak verimli olamaz. Diğer taraftan, kısa kodların esas belleğe yazılması bellek savurganlığına veya zaman kaybına sebep olabilir. Bu yüzden kısa kodların genel amaçlı registerlerden birinde bir birinin yanına yazılmakla uzun kod oluşturulması ve bu kodun belleğe yazılması yoluna gidilmiştir. Açma zamanı yine de genel amaçlı bir register kullanmakla, bu uzun kodlar gereken uzunluklu kısa kodlara parçalanır. Bunu temin edebilmek için aşağıdaki parametrelerden yararlanılmıştır:

d_0 : Bir genel amaçlı registerin bit cinsinden uzunluğu

d_1 : Bir registre yazılabilecek yardımcı bilgilerin toplam bit sayısı

$l(k)$: Bir yardımcı bilginin bit cinsinden uzunluğu

d_2 : Genel amaçlı register → Esas bellek göndermelerinin sayısı

d_3 : Genel amaçlı registerin bayt cinsinden uzunluğu

A_D : Yardımcı bilginin birinci bileşeninin adresi

$G_{2,2}$ 'den sonra G_1^1 'e şartlı atlama yapan $G_{2,3}$ prosedürü gelmelidir. Bu yüzden d_0 ,

d_1 ve d_2 parametrelerinin G_g prosedürü ile önceden belirlenmesi gerekir (Tablo 3.5).

Tablo 3.5. G_g prosedürü

Etiket	İşlem	Açıklama
U1	$l(k) = \langle \text{değer} \rangle$	
U2	$d_0 = \langle \text{değer} \rangle$	

U3	$d_1 = \langle \text{değer} \rangle$	
U4	$A = \langle \text{değer} \rangle$	
U5	SON	G_g prosedürünün sonu

SDY-C için $G_{2,2}$ prosedürü Tablo 3.6'da gösterilmiştir.

Tablo 3.6. $G_{2,2}$ prosedürü

Etiket	İşlem	Açıklama
D1	$P_1 = d_0 - d_1$	Genel amaçlı registerde boş yerin olup olmamasının kontrolü
D2	$P_2 = P_1 - l(k)$	
D3	Eğer $P_2 < 0$, o zaman D4, değilse D10	
D4	$GAR = LP_1(GAR)$	Genel amaçlı registerin içeriği P_1 bit kaydırılır
D5	$[A_D] = GAR$	Genel amaçlı registerin içeriği belleğe yazılır
D6	$d_1 = 0$	GAR → Esas bellek göndermelerinin sayısı
D7	$K_2 = K_2 + 1$	
D8	Eğer $i < N$ ise D9, değilse D13	
D9	$A_d = A_d + d_3$	$G_{3,2}^1$ prosedürüne şartlı atlama
D10	$d_1 = d_1 + l(k)$	
D11	Eğer $i = N$ ise D12, değilse T2	
D12	$P_1 = d_0 - d_1$	GAR'ın boş basamaklarının sayısı
D13	SON	$G_{2,2}$ prosedürünün sonu

Böylece, G_g , $G_{0,1}$ ve $G_{2,2}$ prosedürleri birlikte bir $\bar{s}_i'$ ve bir de K_i hazırlarlar. $\bar{S}_K'$ ve $\bar{D}_K$ serilerinin bileşenlerinin tümünü elde etmek için i 'ye göre döngü prosedürü ($G_{3,2}^2$) organize etmek gerekir (Tablo 3.7).

Tablo 3.7. $G_{3,2}^2$ prosedürü

Etiket	İşlem	Açıklama
T1	$GAR=LI(k)GAR$	GAR'ın içeriği $l(k)$ kadar kaydırılır
T2	$i=i+1$	
T3	Y1'e atla	
T4	SON	$G_{2,3}$ prosedürünün sonu

Böylece, SDY-C yönteminin algoritmik şeması aşağıdaki gibi olur.

$$\overline{SDY-C} = \langle G_g, G_{0,1}^2, G_{2,2}, G_{2,3} \rangle \quad (3.4)$$

3.2. Artışlar yönteminin (AY) algoritması

AİY-C ve SDY-C2 den farklı olarak, artışlar yöntemine (AY) göre, $\bar{s}_i$ kortejinin bileşenleri dolgu sembolleri ile değil $\bar{s}_e$ kortejinin aynı konumlu bileşenleri ile karşılaştırılır. Genellikle, her bir $\bar{s}_i$ için $\bar{s}_e$ olarak $\bar{s}_{i-1}$ veya $\bar{s}_{i+1}$ korteji kullanılır[5,8,11]. Bu durumda uygun olarak $\bar{s}_0$ veya $\bar{s}_{N+1}$ kortejlerinin önceden belirlenmesi gerekir. Yani, buradaki G_1^2 prosedürü yukarıda ele alınmış olan G_1^1 prosedüründen bir Fx2 işlemine sahip olması ile farklanır. Bu işlem ise $\bar{s}_i$ kortejinin bileşenlerini $\bar{s}_e$ kortejinin aynı konumlu bileşenleri ile karşılaştırmak ve şartlı atlama yapmak için kullanılır. Bundan başka G_1^2 prosedürü, her bir $\bar{s}_i$ için $\bar{s}_e$ kortejini belirlemek ve $\bar{s}_0$ kortejini seçmek için ilave olarak Fx2 ve Fx0 işlemlerini de kullanmaktadır. Yukarıda olduğu gibi, burada da G_1^2 prosedürü G_0 prosedürü ile birleştirilerek $G_{0,1}^2$ olarak tanımlanacaktır (Tablo 3.8).

Tablo 3.8. $G_{0,1}^2$ prosedürü

Etiket	İşlem	Açıklama
U1	$i=1$	
U2	$N=<\text{değer}>$	
U2	$A_s=<\text{değer}>$	
U3	$n=<\text{değer}>$	
Fx0	$\bar{s}_0=<\text{değer}>$	
Y1	Okuma	
Y2	$j=1$	
Y3	$r_i=0$	
Fx1	Eğer $a_j^i = a_j^{ie}$ ise Y5, değilse Y8	
Y5	$r_i=r_i+1$	
Y6	$j=j+1$	
Y8	$K_i = n - r_i$	
Y9	$\bar{s}_i' = \text{Pr}(r_i + 1, n) \bar{s}_i$	
Y10	$[A_s] = \bar{s}_i'$	
Y11	$A_s = A_s + K_i$	
Y12	Son	$G_{0,1}^2$ prosedürünün sonu

Görüldüğü gibi, AY-C

$$\overline{AY1-C} = \langle G_g, G_{0,1}^1, G_{1,2}, G_{1,3} \rangle \quad (3.5)$$

durumunda

$$S_p = \langle \langle \bar{s}_i', * \rangle \rangle_{i=1,2,\dots,N}$$

veri yapısını,

$$\overline{AY21-C} = \langle G_g, G_{0,1}^2, G_{2,2}, G_{2,3} \rangle \quad (3.6)$$

durumunda ise

$$S_p = \{ \langle \bar{s}'_i \rangle_{i=1,2,\dots,N}, \langle K_i \rangle_{i=1,2,\dots,N} \}$$

veri yapısını üretecektir.

(3.5) - (3.6) yapıları (3.3)-(3.4) yapıları ile karşılaştırıldığında AIY-C, AY1-C, AY2-C ve SDY-C yöntemleri burada SKY1-C ve SKY2-C olarak adlandırılan iki algoritmadan biri ile gerçekleştirilebilir. Lakin bunun için aşağıdaki şartın yerine getirilmesi gerekir.

$$\left. \begin{aligned} \exists(G_{0,3}^3 = \Pr(S \prec YX - C) \& \exists P(P = 0) : G_{0,1}^3 \equiv G_{0,1}^1 \\ \exists(G_{0,3}^3 = \Pr(S \prec YX - C) \& \exists P(P = 1) : G_{0,1}^3 \equiv G_{0,1}^2 \end{aligned} \right\} \quad (3.7)$$

Burada SKY- soldan kesme yöntemi olarak tanımlanmaktadır [5]. Başka bir deyişle, söz konusu yöntemlerin tek bir algoritma ile gerçekleştirilebilmesi için çıkış veri yapısını tanımlayacak olan bir P değişkenine ihtiyaç vardır.

3.3. AIY-C, SDY-C AY1-C ve AY2-C yöntemleri ile kartejleri kesme algoritmalarının birleştirilmesi

Algoritmaların birleştirilmesi aşağıdaki sebeplere göre önemlidir.

1. Birleşmiş E0 algoritması, tek tek E1, E2, ..., En algoritmalarının toplamından daha az işlem gerektirmektedir.
2. Programların sayısının azaltılması.

Tablo 3.2 -3.8'de görüldüğü gibi, ele alınmış olan algoritmalar ortak kısımlara sahipler. Bu yüzden söz konusu algoritmaları birleştirmekle önemli bir ölçüde

sadeleştirme elde etmek mümkündür ki bu da sonuç da daha az bellek kullanmakla daha büyük problemlerin çözülmesine imkan vermektedir.

Algoritmalar lojik şemalar ve graf şemalarla temsil edilmekle sadeleştirile ve birleştirilebilir[86,88]. [89]'da bu amaçla blokların kullanılması önerilmiştir. Bu yöntemle göre, ayrı ayrı algoritmalar ele bir şekilde bloklara bölünür ki, ortak olan bloklar meydana çıkıyor ve bu bloklar bütün programlar için ortak altprogramlar olarak kullanılır. Bu yöntemi $G_{0,1}^1$ ve $G_{0,1}^2$ prosedürlerine uygulayarak aşağıdaki ortak blokları elde edebiliriz:

$$B_1 = \langle I_i \rangle_{i=1,2,\dots,4}; B_1^2 = \emptyset; B_2^2 = Fx0; B_3^1 = \emptyset; B_3^2 = Fx2; B_4 = \langle Y_i \rangle_{i=1,2,3}; B_5^1 = Fx1;$$

$$B_5^2 = Y4; B_6 = \langle Y_i \rangle_{i=5,6,7,8,9,10,11}$$

Burada yukarı indis prosedürün numarasını, göstermektedir.

Gösterilen blokları yukarı indislerine göre birleştirmekle $G_{0,1}^1$ ve $G_{0,1}^2$ için aşağıdaki birleştirilmiş algoritma elde edilir:

$$G_{0,1}^1 = \{B_1, \emptyset, \emptyset, B_4, Y4, B_6\};$$

$$G_{0,1}^2 = \{B_1, Fx0, Fx2, B_4, Fx1, B_6\}$$

Görüldüğü gibi, $G_{0,1}^1$ algoritması $G_{0,1}^2$ 'de bulunmayan tek bir Y4 işlemine sahiptir. Bu yüzden bu iki algoritmanın birleştirilmesi neticesinde karmaşıklığı aşağıdaki gibi olan bir $G_{0,1}^3$ algoritması elde edilecektir.

$$W(G_{0,1}^3) \cong 0,5(W(G_{0,1}^1) + W(G_{0,1}^2))$$

$G_{0,1}^3$ algoritmasını elde etmek için algoritmaların mantık şemaları yönteminden yararlanalım [90].

$$P = \begin{cases} 0, \text{ eğer gereken } G_{0,1}^1 \text{ ise} \\ 1, \text{ eğer gereken } G_{0,1}^2 \text{ ise} \end{cases}$$

Bu durumda $G_{0,1}^1$ ve $G_{0,1}^2$ 'nin birleşmiş lojik şeması aşağıdaki gibi olacaktır.

$$A_{0,1} = B_1 P \uparrow^1 F_{x0} F_{x2} \downarrow^1 B_4 P \uparrow^2 \omega \uparrow^3 \downarrow^2 Y_4 B_6 \quad (3.8)$$

Burada ω değeri her zaman yanlış (0) olan sahta bir değişkendir.

Şimdi (3.8) şemasının minimal olduğunu göstermek gerekir. Bunun için söz konusu şemaya lojik eşdeğer olan fakat ondan sade olan hiçbir diğer şemanın olamayacağını ispatlamak lazımdır. Bunun için fonksiyonel kompozisyon yaklaşımından yararlanalım[93].

Yukarıda gösterilmiş olduğu gibi, $G_{0,1}^1$ algoritması tek bir işlemin dışında tümü ile $G_{0,1}^2$ algoritması tarafından içermektedir. Bundan ilave Tablo 3.1 ve 3.8' de görüldüğü gibi,

$$(Y_4 = F_{x1}) \longleftrightarrow \forall (i = 1, 2, \dots, N) \forall (j = 1, 2, \dots, n) (a_j^{ie} = DS)$$

$$\forall i \forall j (a_j^{ie} = DS) \longleftrightarrow (F_{x2} = \emptyset) \& \exists Y_{K1} ((Y_{K1} : s_j^{ie} \longrightarrow DS) \& (Y_{K1} < F_{x1})) \quad (3.9)$$

Burada $Fx2 = \emptyset$ şartının karşılanması için $\bar{s}_{ie}$ kerteji, $\bar{s}_{i-1}$ kertejinin işlenmesinin başlangıcında değil, sonunda belirlenmelidir.

$G_{0,1}^2$ prosedürünün yukarıda gösterilen şekilde organize edilebilmesi aşağıdaki ifadeden belli oluyor.

$$\forall (i = 2, 3, \dots, N) (Fx2 : \bar{s}_{ie} = \bar{s}_{i-1}) \equiv \forall (i = 1, 2, \dots, N-1) (Fx2 : \bar{s}_{(i-1)e} = \bar{s}_i)$$

Buradan,

$$\forall (i = 1, 2, \dots, N-1) (Fx2(i+1) \equiv Fx2'(i))$$

Böylece, $\bar{s}_0'$ a ihtiyaç kalmıyor, ancak $Fx2'$ işlemi etalon kertejleri ikinciden başlayarak belirleyebildiğinden, $\bar{s}_1$ için bir $\bar{s}_{1e}$ belirlemek gerekir. Buna uygun olarak $G_{0,1}^2$ prosedürünün $Fx0$ işlemi de aşağıdaki gibi olmalıdır.

$$Fx0 : \bar{s}_{1e} = \langle \text{değer} \rangle$$

$Fx2$ işlemi kullanıldığında kertejlerin etalonlarının indislenmesine de gerek kalmıyor. Böylece,

$$Fx0' : \bar{s}_{1e} = \langle \text{değer} \rangle; Fx2 : \bar{s}_e = \bar{s}_i$$

Şimdi ise (3.9)'un sağ tarafındaki ikinci şartı ele alalım.

$$\exists Y_{K1} ((Y_{K1} : s_i^{ie} \longrightarrow DS) \& (Y_{K1} < Fx1)) \quad (3.10)$$

Y_{K1} işlemi F_{x0} işleminin özel bir şekli olduğu için $\bar{s}_e$ etalonuna n basamaktan ibaret olan istenilen bir değer verilebilir. $Y_{K1} < F_{x1}$ şartına gelince ise Tablo 3.5' de görüldüğü gibi, $F_{x0} < F_{x1}$. Bu yüzden $Y4$ işlemi F_{x1} ve F_{x0} işlemlerin kompozisyonu olarak yerine getirilebilir. Bu söylenilenlere göre,

$$\bar{G}_{0,1}^1 = \langle B_1, F_{x0}, B_4, F_{x1}, B_6 \rangle;$$

$$\bar{G}_{0,1}^2 = \langle B_1, F_{x0}, B_4, F_{x1}, B_6, F_{x2} \rangle$$

Bu algoritmaların birleşmesi ise aşağıdaki gibi olur

$$A_{0,1}^0 = B_1 F_{x0} B_4 F_{x1} B_6 P \uparrow^1 F_{x2} \downarrow^1 a_K \quad (3.11)$$

(3.8) ve (3.11)'in karşılaştırılmasından görüldüğü gibi, $A_{0,1}^0$ mantık şemasının karmaşıklığı $A_{0,1}^1$ mantık şemasının karmaşıklığından bir işlem ve bir dallanma kadar azdır. Bütün bu gelişmelerden sonra elde edilen $G_{0,1}^3$ birleşmiş prosedürü Tablo 3.9'da gösterilmiştir. Bu tabloda gösterilmiş olan U5 işlemi dallanma organize etmek için kullanılmıştır.

Tablo 3.9. $G_{0,1}^3$ prosedürü

Etiket	İşlem	Açıklama
U1	i=1	Giriş verileri
U2	N=< değer>	
U3	As=< değer>	
U4	n=< değer>	

F0'	$\bar{s}_e = \langle \text{değer} \rangle$	
M0	$P = \langle \text{değer} \rangle$	
Y1	$\bar{s}_i$ 'nin okunması	$\bar{s}_i$ kortejinin okunması ve aynı bileşenler zincirinin uzunluğunun belirlenmesi
Y2	$j=1$	
Y3	$r_i=0$	
F1'	Eğer $a_j^i = a_j^e$ ise Y5, değilse Y8	
Y5	$r_i = r_{i+1}$	
Y6	$j=j+1$	
Y7	F1' işlemine git	
Y8	$K_i = n - r_i$	$\bar{s}_i'$ kesilmiş kortejinin uzunluğunun belirlenmesi ve bu kortejin As adresine yazılması
Y9	$\bar{s}_i' = \text{Pr}(r_i + 1, n) \bar{s}_i$	
Y10	$[As] = \bar{s}_i'$	
y11	$As = As + K_i$	bir sonraki kesilmiş kortejin yazılacağı adres
M2	Eğer $P=1$ ise F2, değilse SON	Bir sonraki korte için etalon kortejin belirlenmesi
F2	$\bar{s}_e = \bar{s}_i$	

Böylece, $G_{0,1}^3 \equiv G_{0,1}^2 \longleftrightarrow P=1$ ve $G_{0,1}^3 \equiv G_{0,1}^1 \longleftrightarrow P=0$. Bu ise (3.9) şartının karşılanması demektir. Başka bir deyişle,

$$G_{0,1}^3(P=0) \longrightarrow SKY1-C \equiv AII-C \text{ ve } SKY2-C \equiv SDY-C \quad (3.12)$$

$$G_{0,1}^3(P=1) \longrightarrow SKY1-C \equiv AY1-C \text{ ve } SKY2-C \equiv AY2-C \quad (3.13)$$

Buradan görüldüğü gibi, SKY1-C ve SKY2-C olduğu durumda AIY-C, SDY-C, AY1-C ve AY2C yöntemlerine gerek kalmıyor. Bu tespit söz konusu yöntemlerin esas özelliklerinin verilmiş olduğu Tablo 3.10'da da açıkça görülmektedir.

Tablo 3.10. U_{rel} ve U_{for} tip fazlalıkları gideren yöntemlerin özellikleri

Yöntem \ Özellik	Giderilen fazlalık tipi		Rejim		Sıkıştırılmış veri yapısı
	P=0	P=1	P=0	P=1	
AY1-C	U_{rel}				S_P
AY2-C	U_{rel}				S_P
AIY-C	U_{for}				S_P
SDY-C	U_{for}				S_K
SKY1-C	U_{for}	U_{rel}	AIY-C	AY1-C	S_K
SKY2-C	U_{for}	U_{rel}	SDY-C	AY2-C	S_K

Böylece, 4 değişik yöntem yerine SKY1-C ve SCY2-C olarak tanımlanmış olan sadece 2 yöntemin kullanılması yeterlidir. Bunlara uygun olan açma yöntemlerini ise SKY1-R ve SCY2-R olarak adlandıralım.

3.4. SKY1-C ve SKY2-C yöntemlerinin algoritmalarının birleştirilmesi

Bundan önceki bölümde, prosedürlerin en iyi birleşme şeklini bulmak için işlem etiketlerinin kortajları kullanılmış oldu. Buna benzer olarak, algoritmaları da prosedürlerin kortajları şeklinde ifade etmek olur. Yukarıda ele alınmış olan

$$G_0^1, G_0^2, G_0^3, G_1^1, G_1^2, G_1^3, G_{1,2}, G_{1,3}, G_{2,2}, G_{2,3} \text{ ve } G_g$$

prosedürlerinde aynı işlemler aynı etiketlerle nişanlanmıştır. B ise işlemler arasındaki ilişkilerin işlemlerin yer değişmesinden bakımsız olmasını sağlamaktadır. Böylece, en iyi birleşme için işlemlerin yerini istenildiği gibi değişme imkanı sağlanmıştır. Mesela,

SKY1-C ve SKY2-C yöntemlerinin algoritmaları prosedürlerin aşağıda verilmiş olan ilişkisinden elde edilmektedir.

$$G_0^3 \longrightarrow G_1^3 \longrightarrow G_{1,2} \longrightarrow G_{1,3}$$

$$G_0^3 \longrightarrow G_g \longrightarrow G_1^3 \longrightarrow G_{1,2} \longrightarrow G_{1,3}$$

Bu ilişkiler uygun olarak aşağıdaki kortejlerle verilebilirler:

$$\text{SKY1-C} = \langle G_0^3, G_1^3, G_{1,2}, G_{1,3} \rangle \quad (3.14)$$

$$\text{SKY2-C} = \langle G_0^3, G_g, G_1^3, G_{1,2}, G_{1,3} \rangle \quad (3.15)$$

Bu algoritmaların birleştirilmiş biçimi aşağıdaki gibidir.

$$A = G_0^3 \nu \uparrow^1 G_g \downarrow^1 G_1^3 \bar{\nu} \uparrow^2 G_{1,2} G_{1,3} \omega \uparrow^3 \downarrow^2 G_{2,2} G_{2,3} \downarrow^3 a_K. \quad (3.16)$$

Buna uygun olan prosedürler korteji ise aşağıdaki gibidir

$$\text{SKY-C} = \langle G_0^3, M_1, M_4, G_g G_1^3 M_3 G_{1,2} G_{1,3} G_{2,2} G_{2,3} \rangle$$

Burada, M_1 : $\nu = \langle \text{değer} \rangle$, M_4 : Eğer $\nu = 1$ ise G_g , değilse G_1^3 ; M_3 : Eğer $\nu = 0$ ise G_g , değilse $G_{2,2}$ anlamına gelmektedir. SKY-C yönteminin algoritması Tablo 3.11'de verilmiştir.

Tablo 3.11. SKY-C algoritması

Etiket	İşlem
U1	$i=1$
U2	$N=<\text{değer}>$
U3	$A_s=<\text{değer}>$
U4	$n=<\text{değer}>$
F ₀	$\bar{s}_e=<\text{değer}>$
M ₀	$P=<\text{değer}>$
M ₁	$v=<\text{değer}>$
M ₁	Eğer $v=0$ ise Y1, değilse U _{g1}
U _{g1}	$l(k)=<\text{değer}>$
U _{g2}	$d_0=<\text{değer}>$
U _{g3}	$d_1=<\text{değer}>$
U _{g4}	$d_2=<\text{değer}>$
U _{g5}	$A_D=<\text{değer}>$
Y1	$\bar{s}_i$ 'nin okunması
Y2	$j=1$
Y3	$r_i=0$
F ₁	Eğer $a_j^i = a_j^e$ ise Y5, değilse Y8
Y5	$r_i = r_i + 1$
Y6	$j=j+1$
Y7	F ₁ 'e git
Y8	$k_i = n - r_i$
Y9	$\bar{s}_i' = \text{Pr}(r_i+1, n) \bar{s}_i$
Y10	$[A_s] = \bar{s}_i'$
Y11	$A_s = A_s + k_i$
M ₂	Eğer $P=1$ ise F ₂ , değilse M ₃
F ₂	$\bar{s}_e = \bar{s}_i$
M ₃	Eğer $v=0$ ise Q ₁ , değilse D ₁
Q ₁	$[A_s]=*$

Q ₂	$A_s = [A_s] + 1$
T1	$i = s + 1$
T2	eğer $i \leq N$ ise Y1, değilse T3
T3	SKY1-C'nin SONU
D1	$P_1 = d_0 - d_1$
D2	$P_2 = P_1 - l(k)$
D3	Eğer $P_2 < 0$ ise D4, değilse D10
D4	$REG = LP_1(REG)$
D5	$[A_D] = REG$
D6	$d_1 = 0$
D7	$d_2 = d_2 + 1$
D8	Eğer $i < N$ ise D9, değilse D15
D9	$A_D = A_D + d_3$
D10	$REG = REG \vee K_i$
D11	$d_1 = d_1 + l(k)$
D12	Eğer $i = N$ ise D13, değilse T1.2
D13	$P_1 = d_0 - d_1$
D14	D4'e git
D15	SKY2-C'nin SONU
T1.2	$REG = L1(k)REG$
T2.2	$i = i + 1$
T3.2	Y1'e git

3.5. SKY1-R ve SKY2-R açma yöntemlerinin algoritmaları

SKY1-C ve SKY2-C uygun olarak $S_p = \langle \langle \bar{s}'_i, * \rangle \rangle_{i=1,2,\dots,N}$ ve $S_K = \{ \langle \bar{s}'_i \rangle_{i=1,2,\dots,N}, \langle K_i \rangle_{i=1,2,\dots,N} \}$ sıkıştırılmış veri yapılarını üretirler. Burada, kesilmiş olan $\bar{s}'_i$ karteji $\sqrt[n-r_i]{}$ uzunluğunda olup $\bar{s}'_i = \langle a_j^i \rangle_{j=r_i+1,\dots,n}$ şeklindedir. Bunu dikkate alarak, S_p ve S_K yapıları aşağıdaki şekilde ifade edilebilir:

$$S_p = \langle \langle a_j^i \rangle_{j=r_i+1,\dots,n}, * \rangle \rangle_{i=1,2,\dots,N} \quad (3.18)$$

$$S_K = \{ \langle \langle a_j^i \rangle_{j=r_i+1,\dots,n}, \langle K_i \rangle_{i=1,2,\dots,N} \} \quad (3.19)$$

(3.19) ifadesindeki birinci eleman $\bar{s}'_K$ olarak gösterilirse $S_K = \{ \bar{s}'_K, \langle K_i \rangle_{i=1,2,\dots,N} \}$ olur. $\bar{s}'_K$ kartejinin bileşenleri j ve i 'ye göre indislenemediğine göre yeni bir $g=1,2,\dots,N_K$ indislemesine ihtiyaç meydana çıkmaktadır, yani $\bar{s}_{KX} = \langle a_g \rangle_{g=1,2,\dots,N_K}$, $N_K = \sum_{i=1}^N K_i$. Aynı durum S_p için de geçerlidir. Yani $\bar{s}_{pX} = \langle a_g \rangle_{g=1,2,\dots,N_p}$, $N_p = N_K + N$. Burada N : ayrıştırıcı işaretlerin sayısıdır. Bu yolla biçimlendirilmiş olan $\bar{s}_{pX}$ ve $\bar{s}_{KX}$ sıkıştırılmış veri yapılarından kesilmiş kartejlerin seçilme prosedürleri olan R2.1 ve R2.2 procedürleri, uygun olarak Tablo 3.12 ve 3.13'de verilmiştir.

Tablo 3.13 R2.1 Prosedürü

Etiket	İşlem	Açıklama
Z1	$k_i=0$	Kartejin başlangıç uzunluğu
Z2	$\bar{s}'_i = \emptyset$	Kartejin n başlangıç durumu
Z3	a_g 'yi oku	Kartejin bir bileşeninin okunması

Z4	Eğer $a_g \neq *$ ise değilsen Q1(R2.2 prosedürü)	Bir sonraki ayrıştırıcı işaretin aranması.
Z5	$\bar{s}'_i = \bar{s}'_i \times a_g$	Kortejin oluşturulması
Z6	$k_i = k_i + 1$	Kortejin uzunluğunun belirlenmesi
Z7	$A_s = A_s + 1$	Bir sonraki bileşenin adresinin hesaplanması
Z8	Z4'e git	

Tablo 3.14 R2.2 Prosedürü

Etiket	İşlem	Açıklama
Q1	$P_1 = d_0 - d_1$	REG' in durumunun kontrolü ve on yeni bilgi kısmının yüklenmesi
Q2	$P_2 = P_1 - l(k)$	
Q3	Eğer $P_2 < 0$ ise Q4, değilse Q7	
Q4	$REG = [A_D]$	
Q5	$d_1 = 0$	A_d adresinin yenilenmesi
Q6	$d_0 = \langle \text{değer} \rangle$	
Q7	$A_d = A_d + d_3$	
Q8	$k_i = REG$	
Q9	$d_1 = d_1 + 1$	REG'in içeriğinin l(k) kadar sola kaydırılması
Q10	$REG = L1(k)REG$	
Q11	$A_s' = A_s + k_i - 1$	
Q12	$\bar{s}'_i = Pr(A_s, A'_s) \bar{s}'_{KX}$	

SKY1-R ve SKY2-R için ortak olan parametrelerin başlangıç değerlerini veren bir R_0 prosedürü ve SKY2-R için özel olan parametrelerin değerlerini veren bir R_g prosedürü gerekir (Tablo 3.14 ve 3.15).

Tablo 3.14. R_0 prosedürü

Etiket	İşlem	Açıklama
U1	$i=1$	
U2	$N=<\text{değer}>$	Kortejlerin sayısı
U3	$A_s=<\text{değer}>$	$\bar{S}_{PY}$ veya $\bar{S}_{KX}$ serisinin başlangıç adresi
U4	$n=<\text{değer}>$	Bir kortejin uzunluğu
M0	$P=<\text{değer}>$	SKY'nin çalışma modu
F0	$\bar{s}_e=<\text{değer}>$	Etalon değer

Tablo 3.15. R_g prosedürü

Etiket	İşlem	Açıklama
Ug1	$l(k)=<\text{değer}>$	Sınıflandırma diyagramının bir elemanının bit cinsinden uzunluğu
Ug2	$d_0=<\text{değer}>$	REG'in bit cinsinden uzunluğu
Ug3	$d_1=<\text{değer}>$	REG'in dolu kısmının bit cinsinden uzunluğu
Ug4	$A_d=<\text{değer}>$	Sınıflandırma diyagramının başlangıç adresi

Bölüm 3.3'de gösterildiği gibi, her iki SKY'ye göre de kortejler aynı şekilde $G_{0,1}^3$ prosedürü tarafından kesilir. Bu yüzden de her iki yöntem için de ortak bir R_1 açma algoritması gerekecektir. R_1 ' den sonra ise her iki yöntem için ortak olan ve döngü organize eden bir R_3 prosedürü kullanılacaktır. Tablo 3.16'da bu iki prosedür birlikte verilmiştir.

Tablo 3.16. $R_{1,3}$ prosedürü

Etiket	İşlem	Açıklama
B1	$r_i=n-k_i$	Kesilen bileşenler zincirinin uzunluğu
B2	$\bar{s}'' = \text{Pr}(1, r_i) \bar{s}_e$	Kesilen zincirin elde edilmesi
B3	$\bar{s}_i = \bar{s}_i'' \times \bar{s}_i'$	Kesilen zincirin yerine koyulması

T1	Eğer $i < N$ ise M2, değilse T5	Döngü
M2	Eğer $P=1$ ise F2, değilse T2	Çalışma moduna göre dallanma
F2	$\bar{s}_e = \bar{s}_i$	Etalon hazırlama
T2	$A_s = A_s + 1$	Bir sonraki kesilmiş kortejin adresi
T3	$i = i + 1$	Bir sonraki kortejin indisi
M3	Eğer $v=1$ ise Q1(R2.2 prosedürü), değilse Z1(R2.1) prosedürü	Sıkıştırılmış veri yapısına göre dallanma
T5	SON	$R_{1,3}$ prosedürünün sonu

Böylece , SKY1-R ve SKY2-R algoritmaları uygun olarak aşağıdaki prosedürlerden oluşur:

$$SKY1 - R = \langle R_0, R_{2,1}, R_{1,3} \rangle \quad (3.20)$$

$$SKY2 - R = \langle R_0, R_g, R_{2,2}, R_{1,3} \rangle \quad (3.21)$$

Bu algoritmaların birleştirilmiş lojik şeması ise aşağıdaki gibidir.

$$B = R_0 \vee \uparrow^1 R_g \downarrow^4 R_{2,2} \omega \uparrow^2 \downarrow^1 \downarrow^5 R_{2,1} \downarrow^2 R_{1,3} \vee \quad (3.22)$$

Buna uygun olan prosedürler korteji ise aşağıdaki gibidir.

$$SKY - R = \langle R_0, \nu, R_g, R_{2,2}, R_{2,1}, R_{3,1}, R_{1,3}, \nu \rangle \quad (3.23)$$

Bu algoritma, detaylı olarak Tablo 3.17'de verilmiştir.

Tablo 3.17. SKY-R algoritması

Etiket	İşlem
U1	$i=1$
U2	$N=<\text{değer}>$
U3	$A_s=<\text{değer}>$
U4	$n=<\text{değer}>$
M0	$P=<\text{değer}>$
F0	$\bar{s}_e=<\text{değer}>$
M0	$P=<\text{değer}>$
M1	$v=<\text{değer}>$
M4	Eğer $v=1$ ise U_{g1} , değilse Z1
U_{g1}	$l(k) = <\text{değer}>$
U_{g2}	$d_0 = <\text{değer}>$
U_{g3}	$d_1 = <\text{değer}>$
U_{g4}	$A_d = <\text{değer}>$
Q1	$P_1 = d_0 - d_1$
Q2	$P_2 = P_1 - l(k)$
Q3	Eğer $P_2 < 0$ ise Q4, değilse Q7
Q4	$REG = [A_D]$
Q5	$d_1 = 0$
Q6	$d_0 = <\text{değer}>$
Q7	$A_d = A_d + d_3$
Q8	$k_i = REG$
Q9	$d_1 = d_1 + 1$
Q10	$REG = LI(k)REG$
Q11	$A_s' = A_s + k_i - 1$
Q12	$\bar{s}_i' = Pr(A_s, A_s') \bar{S}_{KX}'$
Q13	B1'e git
Z1	$k_i = 0$
Z2	$\bar{s}_i' = \emptyset$
Z3	$\bar{s}_i'$ 'yi oku

Z4	Eğer $a_g \neq *$ ise değilsen Q1(R2.2 prosedürü)
Z5	$\bar{s}'_i = \bar{s}'_i \times a_g \quad \bar{s}'_i \times a_g$
Z6	$k_i = k_i + 1$
Z7	$A_s = A_s + 1$
Z8	Z4'e git
B1	$r_i = n - k_i$
B2	$\bar{s}'' = \text{Pr}(1, r_i) \bar{s}_e$
B3	$\bar{s}_i = \bar{s}''_i \times \bar{s}'_i$
T1	Eğer $i < N$ ise M2, değilse T5
M2	Eğer $P = 1$ ise F2, değilse T2
F2	$\bar{s}_e = \bar{s}_i$
T2	$A_s = A_s + 1$
T3	$i = i + 1$
M3	Eğer $v = 1$ ise Q1(R2.2 prosedürü), değilse Z1(R2.1) prosedürü
T5	SON

3.6. Birleştirilmiş algoritmalar için başarı değerlendirmeleri

Genellikle, bir algoritmanın tasarlanması sürecinde, onu öyle tertiplemeye çalışırlar ki, onu gerçekleştirecek olan programın komutları sayısı minimum olsun. Bu yüzden, her bir algoritmanın tasarlanması, mutlaka bellek kapasitesi ve gerçekleştirilme süresine karşı koyulmuş olan sınırlamalar dikkate alınarak yerine getiriliyor [3,83]. Bu yüzden algoritmalar için karmaşıklık değerlendirmesi önem kazanmaktadır. Bu problem iki yaklaşımdan birine dayanarak çözülür [3]:

- Algoritmanın kendi karmaşıklığının değerlendirilmesi,
- Algoritmayı gerçekleştiren hesaplama sürecinin değerlendirilmesi

Aşağıda birinci değerlendirme yaklaşımı ele alınacaktır. Çünkü burada hedef olan birleşmiş algoritmalar, birleşmeye katılmış olan algoritmaları gerçekleştirmek için gereken hesaplama süreçlerini her hangi bir şekilde etkilemezler.

Değerlendirme iki kritere göre yapılacaktır: İşlem sayısı ve Gereken bilgi miktarı. Burada değerlendirilecek olan AİY, SDY ve AY yöntemlerinin algoritmalarıdır. Çünkü, bunlar aynı tip fazlalıkları giderir, aynı tip veri yapısı üretir ve aynı fazlalık tiplerini üretirler. Tablo 3.18’de birleştirilmiş algoritmalar için (3.3)-(3.6), (3.14), (3.17), (3.20), (3.21) ve (3.23) ifadelerine göre alınmış olan değerler verilmiştir.

Tablo 3.18. Prosedürlerin ve algoritmaların değersel karakteristikleri

Yöntemler ve prosedürler	Sıkıştırma							Açma		
	AİY	SDY	AY1	AY2	SKY1	SKY2	SKY	SKY1	SKY2	SKY
$G_{0,1}^I=15$	+	+								
$G_{1,2}=2$	+		+		+		+			
$G_{1,3}=3$	+		+		+		+			
$G_{2,2}=15$		+		+		+	+			
$G_g=5$		+		+		+	+			
$G_{2,3}=3$		+		+		+	+			
$G_{0,1}^Z=17$			+	+						
$G_{0,1}^J=20$					+	+	+			
$R_{1,2}=8$								+		+
$R_{2,2}=12$									+	+
$R_0=6$								+	+	+
$R_q=4$									+	+
$R_{1,3}=10$								+	+	+
Toplam	20	38	22	40	25	43	48	24	32	40

Bu tabloda bir prosedürün karmaşıklığı $\langle \text{Prosedür ismi} = \text{değer} \rangle$ şeklinde, bir prosedürün bir algoritma tarafından kullanılması ise + işareti ile gösterilmiştir. En son satırda ise her bir algoritmanın toplam karmaşıklığı verilmiştir. Bu tablodaki değerleri kullanarak algoritmaların birleştirilmesinden elde edilen kazancı belirleyelim. Tabloda gösterilmiş olan AİY-C, SDY-C, AY1-C ve AY2-C yöntemlerinin algoritmalarının toplam karmaşıklığı $20+38+22+40=120$ 'dir. Bunların tümüne eşdeğer olan SKY1-C ve SKY2-C yöntemlerinin algoritmalarının toplam karmaşıklığı ise $35+43=68$ 'dir. Böylece, algoritmaların birleştirilmesi sonucunda %45 kadar kazanç elde edilmektedir. Yukarıda gösterildiği gibi, SKY1-C ve SKY2-C yöntemlerini gerçekleştiren algoritmalar da kolayca birleştirilerek tek bir SKY-C algoritması elde edilebilir. Bu algoritmanın karmaşıklığı 48 olduğuna göre %60 kadar kazanç elde edilmektedir.

3.7. SKY ile veri sıkıştırma oranının değerlendirilmesi

Yukarıda gösterilmiş olduğu gibi, AİY-C, SDY-C, AY1-C ve AY2-C $\bar{s}_i$ kartejini kesilmiş $\bar{s}'_i$ kartejine çeviren dönüştürmelerdir. AİY-C, SDY-C ye göre sıkıştırılacak olan kartejin sol tarafındaki doldurucu semboller zinciri kesilip atılır. AY1-C ve AY2-C'ye göre ise sıkıştırılacak olan kartej etalon bir kartejle karşılaştırılır ve aynı bileşenler zinciri birinci kartejden kesilip atılır. Her iki durumda da $\bar{S} = \{\bar{s}_i\}_{i=1}^N$ serisine $S_n = \{\langle \bar{s}_i, \bar{s}_{ie} \rangle\}_{i=1}^N$ çiftler serisi gibi bakılır. Eğer $\bar{S} = \{\bar{s}_i\}_{i=1}^N$ serisi U_{for} tip fazlalık içerirse o zaman $\forall (i = 1, 2, \dots, N) (\bar{s}_{ie} = \bar{s}_e = \text{const})$. $\bar{S} = \{\bar{s}_i\}_{i=1}^N$ serisi U_{rel} tip fazlalık içerdiğinde ise $\forall (i = 1, 2, \dots, N) (\bar{s}_{ie} = \bar{s}_{i-1})$. Her iki halde SKY-C işlem, $\langle \bar{s}_i, \bar{s}_{ie} \rangle$ çiftleri üzerinde yapar.

Tanım. Eğer $\forall (j < k_i) (\text{Pr}(j) \bar{s}_i = \text{Pr}(j) \bar{s}_{ie})$ ise $\langle \bar{s}_i, \bar{s}_{ie} \rangle > k_i$ dereceden çift olarak tanımlanacaktır.

Böylece, k_i dereceden olan bir çift, uzunluğu k_i olan bir kesilmiş kortej üretecektir.

Bu korteji $s_i^{k_i}$ olarak gösterelim. Bu işaretlemeden sonra SKY-C'nin ürettiği veri yapısı aşağıdaki gibi ifade edilebilir.

$$S_p = \langle \langle s_i^{k_i}, * \rangle \rangle_{i=1}^N \quad (3.24)$$

$$S_K = \{ \langle s_i^{k_i} \rangle_{i=1}^N, \langle K_i \rangle_{i=1}^N \} \quad (3.25)$$

Bu durumda,

$$W(S_p) = N \times W(*) + W(\bar{S}'_K) \quad (3.26)$$

$$W(S_p) = W(\bar{D}_K) + W(\bar{S}'_K) \quad (3.27)$$

$$S'_K = \langle \bar{s}_i^{k_i} \rangle_{i=1}^N, \quad \bar{D}_K = \langle K_i \rangle_{i=1}^N$$

Görüldüğü gibi,

$$N \times W(*) = N \times \left| \log_Q Q \right| = N \quad (3.28)$$

$$W(\bar{D}_K) = \left(N \times \frac{\left\lceil \log_2 n \right\rceil}{\left\lceil \log_2 Q \right\rceil} \right) \quad (3.29)$$

$$W(\bar{S}'_K) = \sum_{i=1}^N K_i \quad (3.30)$$

Burada Q: kullanılmakta olan alfabede bulunan harflerin sayısıdır.

(3.28) ve (3.29) da görüldüğü gibi, n, N ve Q değerlerini kullanmakla $W(*)$ ve

$W(\bar{D}_K)$ değerleri tam olarak hesaplanabilir. Lakin $W(\bar{S}'_K) = \sum_{i=1}^N K_i$ değerinin hassaslıkla

belirlenmesi o kolay değil. Çünkü toplananlar rasgele değerlerdir. Bu yüzden bu değerlerin matematik beklentisi olan $(M(W(\bar{S}'_K)))$ 'yı belirlemeğe çalışalım. Bunun için (3.26) ve (3.27) ifadelerini aşağıdaki şekle getirelim:

$$M(W(S_p)) = N \times W(*) + M(W(\bar{S}'_K)) \quad (3.31)$$

$$M(W(S_K)) = W(\bar{D}_K) + M(W(\bar{S}'_K)) \quad (3.32)$$

Görüldüğü gibi,

$$M(W(\bar{S}'_K)) = m \times M(K_i) \quad (3.33)$$

Burada m: $\bar{S}$ 'de bulunan değişik kortejlerin sayısı, $M(K_i)$: kesilmiş kortejin uzunluğunun matematik beklentisidir. $M(K_i)$ 'yi hesaplamak için aşağıdaki formülden yararlanalım[95].

$$M_i(K_i) = \sum_{K=0}^n l_k \times P_K \quad (3.34)$$

Burada l_k : K_i 'nin mümkün olan değerlerinden biri, P_K ise l_k uzunluklu bir kesilmiş kortejin rastlanması olasılığıdır. P_K 'yı belirlemek için k dereceden çiftlerin rastlanması olasılığını bulmak yeterlidir. Bunun için $\bar{S}$ 'den tekrarlanan kortejlerin tümünü çıkaralım.

$$\bar{S}'' = \{\bar{s}_i''\}_{i=1}^m = \{<a_j^i>_{j=1}^n\}_{i=1}^m$$

Bu seriyi n dereceden seri olarak tanımlayalım. $\bar{S}^n$ serisini a_1^i bileşeninin değerlerine göre alt serilere bölelim. Bu durumda n-1 dereceli seriler elde edilecektir.

$$\hat{S}^{n-1} = \{\bar{S}_{\beta_1}^{n-1}\}_{\beta_1 \in A} = \{\{\langle \beta_1, \langle a_j^i \rangle_{j=2}^n \rangle\}_{i \in M}\}_{\beta_1 \in A}$$

Burada, $M=1,2,\dots,M$. $\bar{S}^{n-1} \in \hat{S}^{n-1}$ serilerini de a_2^i ye bağlı olarak alt serilere bölersek aşağıdaki serileri elde edebiliriz.

$$\hat{S}^{n-2} = \{\{\{\langle \beta_1, \beta_2, \langle a_j^i \rangle_{j=3}^n \rangle\}_{i \in M}\}_{\beta_2 \in A}\}_{\beta_1 \in A}$$

Bu şekilde devam ederek, k. adımda n-k dereceli seriler elde edilir.

$$\hat{S}^{n-k} = \{\dots\{\{\langle \beta_j \rangle_{j=1}^k \langle a_j^i \rangle_{j=k+1}^n \rangle\}_{i \in M}\}_{\beta_k \in A} \dots\}_{\beta_1 \in A} \quad (3.35)$$

(3.35)'de $k=n$ yapmakla her birinde tek bir kertej olan sıfır dereceli seriler elde edilir.

$$\hat{S}^0 = \{\{\dots\{\{\langle \beta_j \rangle_{j=1}^n\}_{\beta_n \in A}\}_{\beta_{n-1} \in A} \dots\}_{\beta_2 \in A}\}_{\beta_1 \in A} \quad (3.36)$$

$|A|=Q$ ve $\beta_i \in A$ olduğu için ve (3.36)'ya dayanarak aşağıdaki ifade yazılabilir.

$$\forall (k=1,2,\dots,n-1) (\bar{S}^{k-1} = \{\Pr(\beta_1, \beta_{n-k}) \bar{S}^k\}_{\beta_{n-k} \in A} \quad (3.37)$$

Böylece:

1. Her bir k+1 dereceli seri en çok Q sayıda k dereceli seri içerebilir.

2. Eğer k dereceli serilerin hiç birisi boş değilse, o zaman $k+1$ dereceli serilerin sayısı $Q^{n-(k+1)}$ dir.

$K=(q>1)$ dereceli serilerin her biri

$$1 < \gamma_q \leq Q \quad (3.38)$$

kadar kortej içermiş olsun. Bu durumda her bir serinin içerisinde $\gamma_q - 1$ adet q dereceli çift oluşacaktır. Her bir $q+1$ dereceli seri en çok Q adet q dereceli seri içerebileceği için $q+i$ dereceli bir serinin içerisinde en çok $Q-1$ adet $q+i$ dereceli çift oluşabilecektir. Bu durumda q ve $q+1$ dereceli çiftlerin frekansı aşağıdaki ifadelerden belirlenebilir.

$$f_q = (\gamma_q - 1) \times Q^{n-q} \quad (3.39)$$

$$f_{q+i} = (Q - 1) \times Q^{n-(q+i)} \quad (3.40)$$

(3.39) ve (3.40) ifadeleri kullanarak $q \leq k \leq n$ dereceli çiftlerin frekans denklemi kurulabilir.

$$(\gamma_q - 1) \times Q^{n-q} + (Q - 1) \times \sum_{k=q+1}^n Q^{n-k} = \alpha \times Q^n - 1 \quad (3.41)$$

Burada $\alpha = \frac{m}{Q^n}$ veri bandının doluluk oranıdır.

γ_q ve α arasında olabilecek sayısal ilişkiyi bulmaya çalışalım.

$$\gamma_q = \alpha \times Q^q \quad (3.42)$$

$$\alpha = \gamma_q \times Q^{-q} \quad (3.43)$$

(3.41)'i (3.43)'de yerine konursak aşağıdakini elde ederiz.

$$(\gamma_q - 1) \times Q^{n-q} + (Q - 1) \times \sum_{k=q+1}^n Q^{n-k} + 1 = \gamma_q \times Q^{n-q} \quad (3.44)$$

(3.44)'ün her iki tarafını $\gamma_q \times Q^{n-q}$ ya bölelim

$$\frac{1}{\gamma_q} \times ((\gamma_q - 1) + (Q - 1) \times \sum_{i=1}^{n-q} Q^{-i} + Q^{-n+q}) = 1 \quad (3.45)$$

Şimdi rasgele bir $\bar{s}_i^{ki}$ kortejinin uzunluğunun matematik beklentisi olan $M(k_i)$ değerini belirleyelim.

Bilindiği gibi, SKY-C'nin k dereceden olan bir çifte uygulanması uzunluğu k olan bir kesilmiş kortej verecektir. Buna dayanarak (3.35) ifadesini $\bar{s}_i^{ki}$ kortejinin uzunluğu için matematik beklentiye çevirelim.

$$M(K_i) = \frac{1}{\gamma_q} \times ((\gamma_q - 1) \times q + (Q - 1) \times \sum_{i=1}^{n-q} Q^{-i} \times (q + i) + n \times Q^{-n+q}) = 1 \quad (3.46)$$

Toplamı açtıktan ve alınmış olan ifadeyi $(Q-1)$ 'e çarptıktan sonra aşağıdaki ifade elde edilir.

$$(Q-1) \times \sum_{i=1}^{n-q} (q+i) \times Q^i + n \times Q^{-n+q} = q+1 + \sum_{i=1}^{n-q-1} i \times Q^{-i} = q + \frac{1}{(1-Q^{-1})^2}$$

Bu ifadeyi (3.46)'da dikkate aldıktan ve basit çevirmelerden sonra aşağıdaki ifade elde edilir.

$$M(K_i) = q + \frac{1}{\gamma_q (1-Q^{-1})^2} \quad (3.47)$$

(3.38.)'i Q^{-q} ye çarpmakla q ve γ_q 'yü belirleyelim.

$$Q^{-q} < \gamma_q \times Q^{-q} \leq Q^{-q+1} \quad (3.48)$$

(3.42)'yi bu ifadede dikkate alarak ve alınmış ifadenin logaritmasını alarak aşağıdaki ifadeyi elde ederiz

$$q > (-\log_Q \alpha) \geq q-1 \quad (3.49)$$

Buradan,

$$q = \lfloor -\log_Q \alpha \rfloor + 1 = \left\lfloor -\log \frac{m}{Q^n} \right\rfloor + 1 = \lfloor n - \log_Q m \rfloor + 1 \quad (3.50)$$

Şimdi (4.42) formülünü kullanarak γ_q 'yü bulabiliriz.

$$\gamma_q = \alpha \times Q^q = m \times Q^{q-n} = m \times Q^{\lfloor n - \log_Q m \rfloor + 1 - n} \quad (3.51)$$

(3.50) ve (3.51)'i (3.46) da dikkate alarak aşağıdaki ifadeyi elde ederiz.

$$M(K_i) = \lfloor n - \log_Q m \rfloor + 1 + \frac{Q^{\log_Q m}}{Q \times m \times (1 - Q^{-1})} = \lfloor n - \log_Q m \rfloor + 1 + \frac{1}{Q-2} \quad (3.52)$$

(3.51)'i (3.33)'te dikkate alırsak

$$M(W(\bar{s}'_k)) = m \times (\lfloor n - \log_Q m \rfloor + 1 + \frac{1}{Q-2}) \quad (3.53)$$

(3.28)'i, (3.29)'u ve (3.53)'ü (3.31) ve (3.32)' de dikkate alarak aşağıdakiler belirlenebilir.

$$M(W(S_p)) = m \times (\lfloor n - \log_Q m \rfloor + 1 + \frac{1}{Q-2}) + N \quad (3.54)$$

$$M(W(S_k)) = m \times (\lfloor \log_Q m \rfloor + 1 + \frac{1}{Q-2}) + \frac{N \times \lceil \log_2 n \rceil}{\lceil \log_2 Q \rceil} \quad (3.55)$$

(3.54)' de görüldüğü gibi, S_p veri yapısında her bir kesilmiş kortej için bir ayrıştırıcı işaret kullanılmaktadır. Bu yüzden, söz konusu durumda, sıkıştırma katsayısının matematik beklentisi aşağıdaki gibi belirlenecektir.

$$M(K_p) = \frac{n}{M(K_i) + 1}$$

Aynısı ile (3.55)'de görüldüğü gibi, S_K veri yapısında her bir kesilmiş kortej için uzunluğu $l(k_i) = \lceil \log_2 n \rceil / \log_2 Q$ olan bir K_i kodu kullanılmaktadır. Bu yüzden, söz konusu durumda sıkıştırma katsayısı aşağıdaki gibi belirlenecektir.

$$M(K_K) = \frac{n}{M(K_i) + l(k_i)}$$

(3.54) ve (3.55)'de görüldüğü gibi, S_P ve S_K veri yapılarının büyüklük farkı yalnız yardımcı bilgilerin farklı olmasından kaynaklanmaktadır. Bu yüzden, söz konusu yapıların hangisinin hangi durumlarda daha az veri miktarı gerektireceğini belirliyelim.

$$N \times W(*) \geq W(D_K) \longleftrightarrow N \geq N \times \frac{\lceil \log_2 n \rceil}{\lceil \log_2 Q \rceil}$$

Buradan,

$$(N \times W(*) \geq W(D_K) \longleftrightarrow \frac{\lceil \log_2 n \rceil}{\lceil \log_2 Q \rceil} \leq 1 \equiv N \times W(*) \geq W(D_K) \longleftrightarrow Q \geq n$$

Yani $n < Q$ olduğunda S_K yapısı, diğer durumlarda ise S_P yapısı daha tasarrufludur.

3. Bölümün sonuçları

1. Ayırıştırıcı işaretler yöntemi(AİY), Sınıflandırıcı diyagramlar yöntemi(SDY) ve Artışlar yöntemi (AY) için algoritmalar tertiplenmiş, bu algoritmalar kesin sonuçlar veren prosedürlere bölünmüştür. AİY ve SDY yöntemlerinin incelenmesinden belli olmuştur ki, onları gerçekleştiren algoritmaların kortej kesme kısımları tamamı ile bir birinin aynısıdır. Bu yöntemler yalnız yardımcı bilgilerin yapısı ile farklılıklar ki, bu da sonuçta sıkıştırılmış olan veri yapılarının farklı olmasına getirip çıkarır.
2. Artışlar yönteminin değişik şekilde gerçekleştirilmesi algoritmalarının incelenmesi, bu yöneme göre kortej kesmenin diğer iki yöntemden yalnız iki ek işlemle farklı olduğunu göstermiştir. Bu yüzden bu üç algoritmanın birleştirilmesi sayesinde elde edilmiş olan algoritmanın kaynak algoritmalarla aynı derecede etkin olduğu gösterilmiştir. Söz konusu birleşmiş algoritma etalon seçme yöntemine göre SKY1-C ve SKY2-C olarak adlandırılmış ve Urel ile Ufor (görelî ve biçimsel) tip fazlalıkları gidermek için terk araç olarak önerilmiştir. Bu yöntemlerin uygun olarak S_p ve S_k tip sıkıştırılmış veri yapısı ürettikleri tespit edilmiştir.
3. Ele alınmış olan yöntemler için karmaşıklık kriteri belirlenmiş, algoritmaların mukayeseli analizi yapılmış ve birleşmiş algoritmanın karmaşıklığının, kaynak algoritmaların toplam karmaşıklığından %43-%45 daha az olduğu gösterilmiştir.
4. SKY1 ve SKY2 yöntemlerinin algoritmalarının da bir SKY algoritması şeklinde birleştirilebileceği gösterilmiş ve bu yolla %58 sadeleştirme elde edilmiştir.
5. Elde edilmiş olan tek birleşmiş algoritmanın fonksiyonel olarak 4 adet müstakil kaynak algoritmaya eşdeğer olduğu gösterilmiş ve bu algoritmaya uygun olarak tek bir veri açma algoritması tertiplenmiştir. Değerlendirmeler göstermiştir ki, bu algoritma müstakil kaynak algoritmalarından %57 daha sadedir.

4. GELİŞTİRİLMİŞ OLAN VERİ SIKIŞTIRMA VE HATASIZ OLARAK AÇMA YÖNTEMLERİNİN SANAYİ MÜESSESİLERİNİN PASAPORTLARINI OTOMATİK OLARAK DÜZENLEYEN VE İŞLEYEN SİSTEMDE UYGULANMASI

4.1. Sanayi müesseselerinin pasaportlarını otomatik olarak düzenleyen ve işleyen sistemde veri tabanının fizik organizasyonu

Sanayi müesseselerinin pasaportlarının otomatlaştırılmış kontrol sisteminin veri tabanında esas veri kütlesi olarak sanayi müessiselerinin pasaportları bulunmaktadır. Bir pasaport, her biri 30 altbölüm içeren 9 bölümden oluşmaktadır. Bütün pasaportların aynı altbölümü aynı bir dosyada toplanmıştır. Bu sistemde aşağıdaki dosya tipleri kullanılmaktadır [73].

$$1. F_j = \{s_{ij}\}_{i=1}^4\}_{j \in \{1,2,3\}}$$

Burada F_1 , F_2 ve F_3 uygun olarak, “Bakanlık”, “Alan” ve “ Bölge” dosyaları, s_{1j} , s_{2j} ve s_{3j} dosyanın şifresi, ismi ve mnemokodudur, s_{4j} “Müessese” dosyasındaki birinci kayda olan göstergedir.

$$2. H_k = \{< h_{ik} >_{i=1}^3\}_{k \in \{1,2,3,4\}}$$

Burada H_1 , H_2 , H_3 ve H_4 uygun olarak, “Müessese”, “ Memul”, “Değer” ve “Birim”. dosyaları, h_{1k} , h_{2k} , ve h_{3k} ise dosyanın şifresi, ismi ve mnemokodudur.

$$3. P = \{< h_i, < s_{ij}, P_{ij} >_{j=1}^3, < r_{iq} >_{q=1}^{30} >_{i=1}^N\}$$

Burada P müesseselerin listesi, h_i müessesenin şifresi, s_{11}, s_{12}, s_{13} ise bakanlığın, alanın ve arazinin şifresidir. P_{11}, P_{12}, P_{13} söz konusu dosyada bakanlığa, alana ve araziye göre bir sonraki kayıtlara olan göstergelerdir. r_{iq} ise R_q dosyasının birinci kaydına olan göstergedir.

$$4. \quad R_q = \{ \{ h_{nk} \}, < a_{ql} >_{l=1}^{mq} \}_{q=1}^{30}$$

Burada h_{nk} , H_k dosyasındaki n numaralı kayda olan göstergedir, a_{ql} ise q bölümünde olan bir niteliktir.

Böylece, ele alınmakta olan veri tabanı 38 dosyadan oluşmaktadır. Bu dosyalardan 7'si (F_1-F_3 ve H_1-H_4) sorgu bilgisidir. Sekizinci dosya P, arama motoru için bilgi tabanıdır ve yalnız dosyalar arası ilişki bilgilerini içermektedir. Dokuzuncu dosya R_1 de sorgu dosyasıdır ve müesseseler hakkında genel bilgiler içermektedir. Geriye kalan 29 dosya (R_2-R_{30}) esas bilgileri ve H_1-H_3 ve P dosyaları ile ilişkileri içermektedirler.

R_2-R_{30} dosyalarının tümü H_1 dosyasına göstericiye sahiptir, ancak bu dosyalardan her biri ilave olarak H_2 , yada H_3 dosyasına gösterici bulundurmaktadır. R_2-R_{30} dosyalarındaki kayıtların tümü değil, büyük bir çoğunluğu H_4 dosyasına da göstericiye sahiptir.

Sanayi müesseselerinin pasaportları 6 basamaklı sayılarla şifrelendiği için H_1-H_3 ve P dosyalarına olan her bir gösterici için 4 baytlık bir alan ayrılmıştır. H_4 dosyasına gösterici içinse yalnız bir bayt ayrılmış durumdadır. Böylece, R_2-R_{30} dosyalarındaki her bir kayıt, esas bilginin yanı sıra, 8-9 baytlık bir ilişki alanı da kullanmaktadır. Ele alınmakta olan sistemde bu alanlarda bulunan bilgiler Asosyatif bilgiler olarak tanımlanmaktadır.

4.2. Hedef veri tabanında veri sıkıştırması

Sanayi müesseselerinin pasaportlarının otomatlaştırılmış kontrol sisteminin şimdiki (yıl 1983) uygulama aşaması F_1 tek bir bakanlık (Petrol işleme ve petrol-Kıma sanayisi bakanlığı) kapsamaktadır. Bu yüzden veri miktarı az olduğu için bu dosyaya sıkıştırma uygulanmadı. İleride, bu dosyaya diğer bakanlıklar da eklendiğinde, bu dosyanın nitelikler listesi genişlenecek, onların değerler kütlesi çoğalacak ve o zaman bu dosyada değişik fazlalık tipler meydana çıkacaktır. Halelik ise, söz konusu dosyada U_{TD} tip fazlalığın meydana çıkacağına kesin bir gözle bakılmaktadır ki, bu da Kodlama yönteminin uygulanmasını gerektirebilir. Aynı sebeplerden dolayı F_2 ve F_3 dosyaları da sıkıştırmaya tabi tutulmamıştır.

H_1 dosyası (Müessese), müesseselerin adlarının listesinden ibarettir. Bu adlar ilk iki sözü aynı olan alt listelere bölünmüştür. Mesela, makinecilik ve petrol imalatı alanında çalışan müesseselerin adları gibi. Bunun dışında, alt listelerdeki kayıtlar uzunluklarına göre bir birinden çok farklılanmaktadırlar. İncelemeler sayesinde, bu listede %42 U_{for} tip ve %21 U_{rel} tip fazlalık olduğu tespit edilmiştir. Söz konusu listede bulunan tüm fazlalık tiplerinin toplam oranı ise %73 olarak tespit edilmiştir. Bu yüzden H_1 dosyasına, gösterilen fazlalıkların tümünü giderecek olan iki taraflı kesme yöntemi uygulanmış ve sıkıştırma katsayısı 3.3 olmuştur.

H_2 dosyası memulların adları listesinden ibarettir. Bu dosya, bir müessese için memul olan bir eşyanın başka bir müessese için kullanılacak malzeme olabileceği olgusu dikkate alınarak tertiplenmiştir. Sonuçta, “Memullar” ve “Kullanılan malzemeler” dosyalarının ortak kısımları olduğu ve bu kısımların ayrıca alt dosyalar olarak tertiplenmesine karar verilmiştir. Bu yolla, U_{OD} tip fazlalığın oluşması önlenmiş ve uygun yöntemin uygulanmasına da gerek kalmamıştır.

“Memullar” dosyasının elemanları, uzunluklarına göre bir birinden çok farklılık göstermektedirler. Bu ise büyük bir oranda U_{for} tip fazlalığın meydana çıkmasına sebep olmuştur. İlave olarak, bu listede bulunan adlar alt listeleri, bir birinden yalnız son bir kaç literalle farklılık göstermektedirler ki, bu da büyük bir oranda U_{rel} tip fazlalığın meydana çıkmasına sebep olmuştur. Bu durumda, istatistik fazlalığın biçimsel fazlalığa getirilebilmesi için bazı adlarda sözlerin yerleri değiştirilmiş, fakat bu iş her hangi semantik(anlamsal) hataya yol açmayacak şekilde yapılmıştır. Bütün bu ön hazırlıklardan sonra, her bir kaydı 80 bayt olan H_2 dosyası incelenmiş ve %62 U_{for} tip ve %9 U_{rel} tip fazlalığa sahip olduğu tespit edilmiştir. Bu dosyaya da iki taraflı kesme yöntemi uygulanmış ve 3.1 kadar bir sıkıştırma katsayısı elde edilmiştir.

H_3 dosyası değişik ekonomik değerlerin adlarından ibaret olan bir listeyi içermektedir. İnceleme sayesinde bu dosyanın %3.5 U_{rel} tip ve %48 U_{for} tip fazlalıklara sahip olduğu belirlenmiştir. Burada bulunan U_{rel} tip fazlalığın düşük olması ve H_3 dosyasının yoğun kullanım görmesi (çok yoğun olarak kullanılmakta olan 21 dosyadan bu dosyaya referans var) ~~ile yoğun ilişkisi var~~, bu dosyaya iki taraflı kesme yönteminin sadece U_{for} tip fazlalığı giderecek olan hızlı modifikasyonu uygulanmış, sıkıştırma katsayısı ise 1.8 olmuştur.

P dosyası “Müesseseler listesi” olarak adlandırılmış ve kayıtlarının elemanlarının tümü göstericilerden ibarettir. Bu dosya incelenerek, pratik olarak hiç bir fazlalık içermiyor sonucuna varılmıştır.

Şimdi de R_1 - R_{30} dosya grubunu ele alalım.

R_1 dosyası müesseselerin adı, bakanlıkların şifresi, alanların şifresi, arazilerin şifresi, yöneticilerin adı, soyadı, adresler vs. den ibaret olan genel bilgiler içermektedir. İncelenme sonucunda, bu dosyada yalnız kötü kodlamadan dolayı meydana çıkan ve tipik olmayan fazlalıkların mevcut olduğu görülmüştür. Bu dosyada kullanılan alfabe, ancak 48

literal kullanmakta, bunun her biri içinse bir baytlık bir alan ayrılmaktadır ki bu da %25 fazlalığın meydana çıkmasına sebep olmuştur. Fakat belleğin baytlarla adreslendiği çağdaş bilgisayarlarda karmaşık haritalama yöntemleri kullanmadan bu fazlalıktan kaçınmak mümkün olmadığına ve söz konusu yöntem performansı önemli bir ölçüde düşürdüğüne göre bu dosyaya biçimsel sıkıştırma uygulanmamıştır. Ancak burada önemli bir ölçüde istatistik fazlalık olduğu için özel olarak bu dosya için geliştirilmiş olan bir kodlama yöntemi kullanılmış, sonuçta %30 sıkıştırma elde edilmiştir [73].

R_2 - R_{30} dosyalarının farklandırıcı niteliği, iki kısımdan ibaret olan kayıtlar kullanmalarıdır. Bunlar göstericiler ve onlu veriler kısımlarıdır. Bu kayıtların sayısal kısımlarında U_{for} ve U_{rel} tip fazlalıklar bulunmaktadır. Fakat bu bölüm vektör (kortej) tip domenlerden organize edildiği için, veriler söz konusu fazlalıklar tam olarak giderebilecek şekilde yer değişme işlemine tabi tutulamamış, bundan dolayı yalnız U_{for} tip fazlalığın giderilmesi için soldan kesme yöntemi uygulanmıştır. Bu durumda kayan etalon seçilemeyeceğinden dolayı bütün değerler için ortak olan, yalnız sıfırlardan ibaret olan tek bir etalon kortej kullanılmıştır.

Sıkıştırma işlemine tabi tutulan veri tabanında bulunan dosyaların esas karakteristikleri Tablo 4.1'de gösterilmiştir. Burada esas ve yardımcı bellekler arasında veri alışverişi için birim olarak 1340 (System IBM360/370) baytlık bloklar kullanıldığı için sıkıştırma sonuçları da bu birimler baz alınarak gösterilmiştir. Bu yüzden sıkıştırma katsayısı ortalama olarak $951134/472937 = 2.05$ olduğuna rağmen, bellek kapasitesi tasarrufu (etkin sıkıştırma katsayısı) yalnız $730/370 = 1.97$ olabilmektedir.

Tablo 4.1. Sanayi Müesseselerinin pasaportlarının otomatlaştırılmış kontrol sisteminin veri tabanındaki dosyaların sıkıştırmadan önceki ve sonraki karakteristikleri.

Dosya	Büyükülüğü (bayt cinsinden)	Blundur- duğı faz- lalık (%)	Sıkıştırma yöntemi	Sıkıştırma- dan sonraki büyükülüğü	Blok sayısı	
					Sıkıştırma- dan önce	Sıkıştırma- dan sonra
H ₁	24000	73	İTKY	720	2	1
H ₂	130240	71	İTKY	45584	98	34
H ₃	12480	48	İTKY	6864	10	6
R ₁	6636	30	KY	4706	5	4
R ₂	85120	57	SKY	35048	64	27
R ₃	23520	33	SKY	16464	18	13
R ₄	13730	45	SKY	6860	11	6
R ₅	68228	62	SKY	28291	51	22
R ₆	37762	30	SKY	22264	24	17
R ₇	5479	42	SKY	3510	5	3
R ₈	86850	46	SKY	43980	65	33
R ₉	6300	26	SKY	4410	5	4
R ₁₀	28728	44	SKY	17243	22	13
R ₁₁	9246	44	SKY	5554	7	5
R ₁₂	19600	47	SKY	10800	15	9
R ₁₃	40300	55	SKY	20850	31	16
R ₁₄	23540	48	SKY	12979	18	10
R ₁₅	10512	55	SKY	5160	8	4
R ₁₆	21820	48	SKY	12200	17	10
R ₁₇	38976	63	SKY	16000	30	12
R ₁₈	52920	50	SKY	27476	40	21
R ₁₉	12062	36	SKY	8342	10	7
R ₂₀	80647	60	SKY	33305	61	25
R ₂₁	30016	58	SKY	13281	23	10
R ₂₂	12270	25	SKY	9766	10	8

R ₂₃	20188	35	SKY	14120	16	11
R ₂₄	13672	35	SKY	9162	11	7
R ₂₅	2800	22	SKY	2310	3	2
R ₂₆	15116	22	SKY	12108	12	10
R ₂₇	5760	47	SKY	8060	5	3
R ₂₈	25572	48	SKY	13280	20	10
R ₂₉	4200	56	SKY	2120	4	2
R ₃₀	10444	53	SKY	5130	9	5
Toplam	951134	54		472937	730	370

4.3. Verileri sıkıştırma ve açma programları

Üçüncü bölümde gösterildiği gibi, SKY yöntemi $S_p = \langle \langle \bar{s}_i', * \rangle \rangle_{i=1}^N$ veya $S_K = \{ \langle \bar{s}_i' \rangle_{i=1}^N, \langle K_i \rangle_{i=1}^N \}$ sıkıştırılmış veri yapısını üretiyor. Tablo 3.11'de bu yöntemin algoritması verilmiştir. Bu tablodan görüldüğü gibi, S_K yapısının oluşturulması S_p yapısına nispeten yaklaşık olarak 2 kat daha çok zaman gerektirir. Fakat S_K yapısının açılması, S_p yapısının açılmasından 2-3 kat daha çabuk olarak gerçekleştirilebilir. Çünkü, S_p yapısı için ayrıştırıcı literallerin bulunması için gerekli olan karakter- karakter inceleme S_K yapısında yoktur ve burada kesilmiş kortej tümü ile seçilip incelenebilir. Bu ise sanayi müesseslerinin pasaportlarını bulunduran veri tabanı için önemli bir olgudur. Çünkü burada verilerin değişme yoğunluğu, onların kullanılma yoğunluğundan karşılaştırılmayacak derecede düşük olduğundan açma yoğunluğu da sıkıştırma yoğunluğundan karşılaştırılmayacak kadar yüksektir. Bu yüzden söz konusu veri tabanı için SKY'nin S_K tip veri yapısı üreten varyantı tercih edilmiştir.

İkinci bölümde gösterildiği gibi, İTKY (iki taraflı kesme yöntemi) $S_{DY} = \{ \langle \vec{s}_i' \rangle_{i=1}^N, \langle r_i \rangle_{i=1}^N, \langle K_i \rangle_{i=1}^N \}$ veri yapısını üretmektedir. Bu yapının açılışı ise S_K yapısının açılışından onunla farklanır ki, r_i bilgisi yardımcı bilginin uygun alanından okunup etalondaki ile aynı olan sağ zincirin oluşturulması için kullanılır. Bu özellik dikkate alınmakla, İTKY-C ve İTKY-R için gerçekleştirilme süreleri Tablo 3.11 ve 3.17 'yi kullanmakla hesaplanabilir.

Tablo 3.11 ve 3.17'de verilmiş olan işlemlerden her biri IBM360/370 assemblerinin 1-3 komutu ile gerçekleştirilebilir. Fakat burada kullanılan işlemlerin büyük çoğunluğu bir assembler komutu ile gerçekleştirilebilen kısa işlemlerdir. Bu özellikler dikkate alınmakla bütün programlar için zaman şemaları [94] kurulmuş, ve bunlar şebeke şekline getirilmiştir[96,97]. Bu şebekeler kullanılarak SKY-C, İTKY-C, SKY-R ve İTKY-R algoritmaları ile bir literalın işlenme süresi için, uygun olarak, aşağıdaki ifadeler elde edilmiştir.

$$t_{1c} = t_{ko} \times (7 + 2/K_c + 8/n) \quad (4.1)$$

$$t_{2c} = t_{ko} \times (7 + 2/K_c + 11/n) \quad (4.2)$$

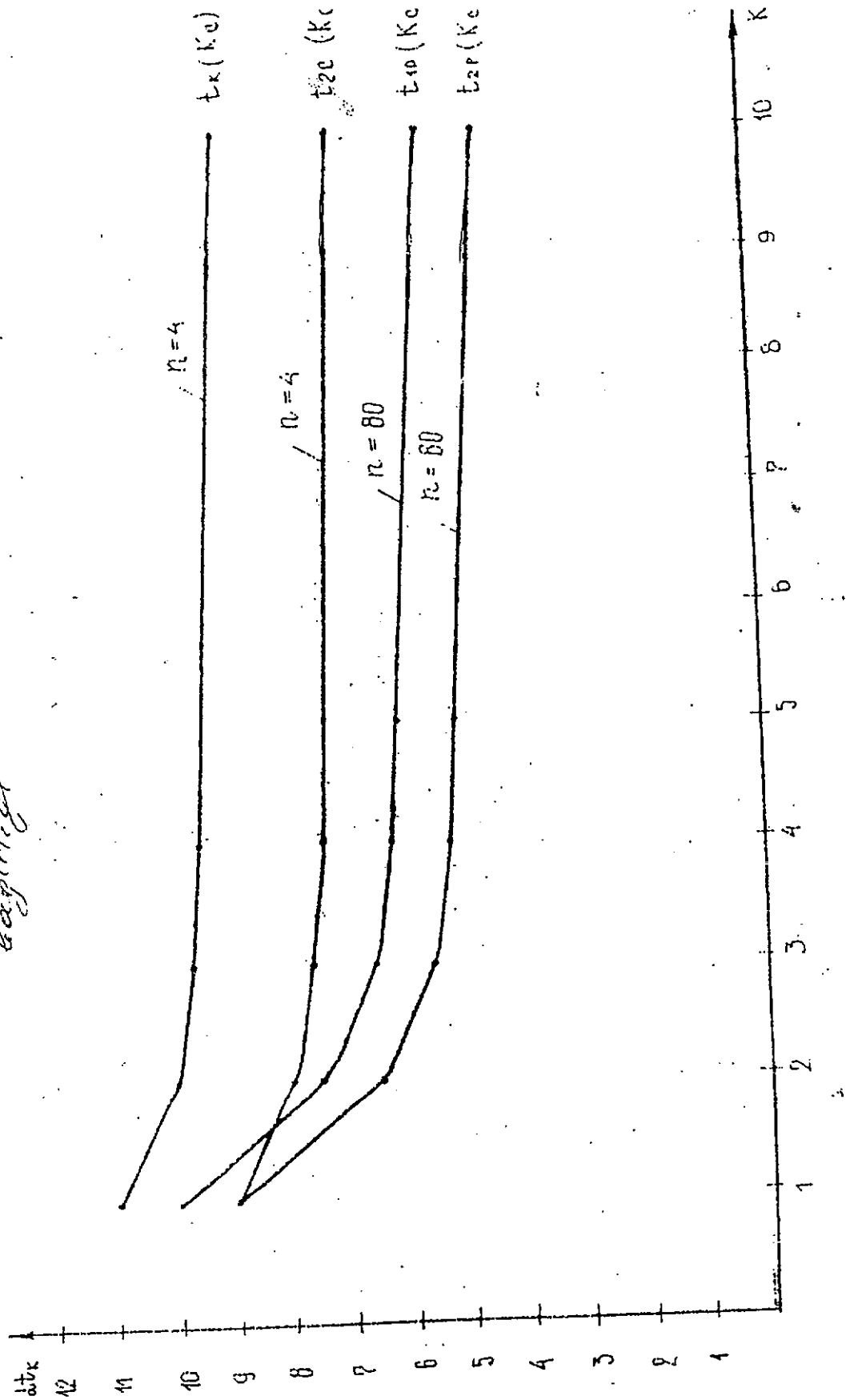
$$t_{1R} = t_{ko} \times (3 + 5/K_c + 8/n) \quad (4.3)$$

$$t_{2R} = t_{ko} \times (4 + 5/K_c + 11/n) \quad (4.4)$$

Burada, n : değerlerin literal cinsinden uzunluğu, t_{ko} : bir kısa işlemin sürdürülme süresi, K_c : sıkıştırma katsayısıdır. Görüldüğü gibi, söz konusu süreler K_c ve n 'e bağlıdır. Şekil 4.1'de n 'in sabit bir değerinde $t_{1c}(K_c)$, $t_{2c}(K_c)$, $t_{1R}(K_c)$ ve $t_{2R}(K_c)$ eğrileri gösterilmiştir.

Gösterilen özellikler dikkate alınarak assembler (IBM360/370) dilinde tertiplenmiş olan programlar sanayi müesseselerinin pasaportlarının otomatlaştırılmış kontrol sisteminin kütüphanesine dahil edilmiştir. PROG1, PROG2, PROG3 ve PROG4 olarak

Şekil 4.1. Bir literali iletme süresinin serbestinme katsayısına
bağımlılığı



adlandırılmış olan bu programların uzunlukları uygun olarak 175, 140, 173 ve 102 komu olup, bellekte 464, 436, 460 ve 224 bayt yer tutmaktadırlar. Söz konusu programlar Sovyetler Birliği Başarılar Sergisi'nde gösteriye sunulmuş ve madalyaya layik görülmüştür.

Onlu sayıları donanımsal olarak sıkıştırma ve açma için geliştirilmiş olan cihaz icat olarak Patent sertifikasına laik görülmüştür.¹

4.4. Sıkıştırma yöntemlerinin uygulanmasından elde edilen ekonomik tasarruf

1., 2. ve seviye bellekler arasında dağıtılmış olan verilerin sıkıştırmadan önceki miktarı uygun olarak V_1 , V_2 ve V_3 , sıkıştırmadan sonraki miktarı ise V'_1 , V'_2 ve V'_3 olsun. Tabii ki belleği daha tasarruflu kullanabilmek için sıkıştırılmış olan verilerin seviyeler üzere yeniden dağıtılması gerekir. Bu ise aşağıdaki şartları karşılamakla yerine getirilebilir.

$$V_1 + W_{2,1} = K_C \times V_1 \quad (4.5)$$

$$V_2 + W_{3,2} = K_C \times V_2 \quad (4.6)$$

$$V_3 - W_{3,2} + A = K_C \times V_1 \quad (4.7)$$

Burada, $W_{(i+1),i}$: (i+1) seviyesinden i seviyesine kaydırılan veri miktarı, A: 3. seviyede söz konusu verilerin bulunmadığı alandır. (4.5)-(4.7) denklemlerinin birlikte çözümlenmesi sonucu aşağıdaki gibidir:

$$W_{2,1} = V_1 \times (K_C - 1) \quad (4.8)$$

$$W_{3,2} = (V_1 + V_2) \times (K_C - 1) \quad (4.9)$$

Sıkıştırma öncesi ve sonrası 1., 2. ve 3. seviyelere başvuru olasılıkları uygun olarak P_1, P_2, P_3 ve P'_1, P'_2, P'_3 olsun. Bu durumda birinci olasılıkları bilerek ikinciler aşağıdaki şekilde bulunabilir:

$$P'_1 = P_1 + \frac{W_{2,1}}{V_2} \times P_2 = P_1 + \frac{V_1 \times (K_C - 1)}{V_2} \times P_2 \quad (4.10)$$

$$P'_2 = P_2 + \frac{W_{2,1}}{V_2} \times P_2 + \frac{W_{3,2}}{V_3} \times P_3 = P_2 + (K_C - 1) \times \left(\frac{V_1 + V_2}{V_3} \times P_3 + \frac{V_1}{V_2} \times P_1 \right) \quad (4.11)$$

$$P'_3 = P_3 - \frac{V_1 + V_2}{V_3} \times (K_C - 1) \times P_3 \quad (4.12)$$

Bu denklemleri kullanarak, sıkıştırma sayesinde elde edilebilecek olan zaman tasarrufunu hesaplayalım.

Yukarıda söylenildiği gibi, söz konusu veri tabanı IBM360-55 makinesinin manyetik disklerinde saklanmaktadır. Bu makinede esas bellek birinci, manyetik disk ise ikinci seviyede olduğundan ve üçüncü seviye bellek olmadığından $P_3 = 0$. Bu yüzden (4.10) ve (4.11) ifadeleri aşağıdaki şekle geliyor:

$$P'_1 = P_1 + \frac{V_1 \times (K_C - 1)}{V_2} \times P_2 \quad (4.13)$$

$$P'_2 = P_2 \times \left(\frac{V_1}{V_2} \times (K_C - 1) \right) \quad (4.14)$$

Yukarıda söylenildiği gibi, veri tabanı toplam olarak 730 bloktan ibarettir ki bunlardan yalnız 73 blok aynı zamanda esas bekte bulunabilir. Bu ise veri tabanının %10'u demektir. Böylece, geriye kalan 637 bloktan her biri yalnız gerektiğinde esas belleğe getirilir ve bunun için esas bellekte bulunmakta olan bloklardan biri çıkarılıp yardımcı belleğe gönderilir. Bütün bu işleri, kullanıcıya sezdirmeden, sanal bellek yönetimi sistemi yerine getirir. Böylece, $V_1=73$, $V_2=730$, $K_C=2$, $P_1=0.1$, $P_2=0.9$. Bu durumda bir bloğa ulaşma süresi aşağıdaki gibi belirlenebilir.

$$T_{ge} = P_1 \times t_{es} + P_2 \times t_{MD} \quad (4.15)$$

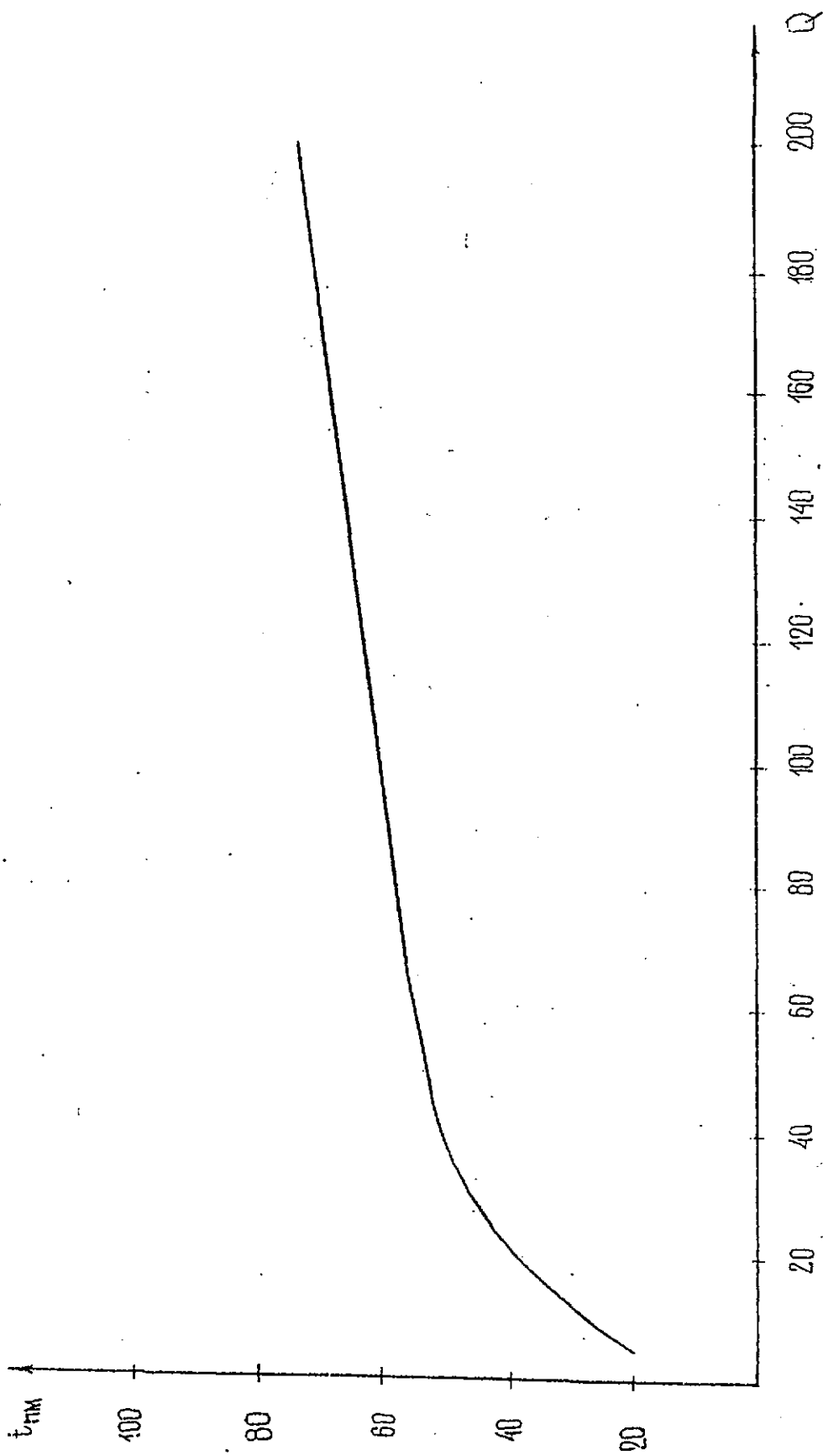
Burada t_{es} , ve t_{MD} , uygun olarak, bir bloğun esas bellekten ve yardımcı bellekten alınma süresidir.

Bilindiği gibi [98,99], $t_{MD} = t_{BY} + t_{YD} + t_{YO}$, burada t_{BY} : başlığın gereken silindire kadar yer değişme süresi, t_{YD} : disklin ortalama olarak yarım devri için gereken süre, t_{YO} ise bir bloğu yazma/okuma süresidir. EC-5061 diski (Sovyetler Birliği) için $t_{YD}=12.5ms$. Bir izde 5 blok yerleştiğine göre $t_{YO}=5ms$. t_{BY} 'ye gelince ise onun değeri verilerin yerleştiği disk bölgesinin genişliğine bağlıdır [87] ve bu amaçla çizilmiş olan bir eğriden belirlenebilir(Şekil 4.2). Şekil 4.2'de görüldüğü gibi, $Q=8(730 \text{ blok } 8 \text{ silindirde yerleşir})$ için $t_{BY}=26.5ms$. Böylece, $t_{MD} = t_{BY} + t_{YD} + t_{YO} = 26.5 + 12.5 + 5 = 44ms$. Bunları (4.15)'de dikkate alırsak,

$$T_{ge} = P_1 \times t_{es} + P_2 \times t_{MD} = 0.1 \times 0 + 0.9 \times 44 = 39.6ms$$

$V_1=73$, $V_2=730$ ve $K_C=2$ olduğunu bilerek (4.13)'ü ve (4.14)'ü kullanmakla verilere sıkıştırmadan sonraki başvuru olasılıkları da hesaplanabilir:

Şekil 4.2. Başlıpın yerleşime göre alınmış
grecilipine bagli



$$P'_1 = P_1 + \frac{V_1 \times (K_C - 1)}{V_2} \times P_2 = 0.19$$

$$P'_2 = P_2 \times \left(\frac{V_1}{V_2} \times (K_C - 1) \right) = 0.81$$

Bu durumda, sıkıştırılmış olan veriler yalnız 4 silindirde yerleşirler. Bu ise manyetik diskte gereken ize ulaşım süresini 26.5ms'den 21.6ms'ye düşürmektedir. Yani $t_{MD} = 44\text{ms}$ 'den 39.1 ms düşecektir. Ancak $K_C=2$ olduğu için bir defaya bir yerine 2 bloğun transferi yapılacaktır. Bu yüzden bir bloğa ulaşım süresi 39.6ms'den 19.5ms kadar düşecektir. Lakin burada bir olumsuzluk da var. Esas belleğe sıkıştırılmış veriler getirileceğinden dolayı bunların açılması için ek bir t_R süresi gerekecektir. Bu yüzden normal veri biçimine ulaşım için gereken tam süre aşağıdaki gibi belirlenecektir.

$$T_{tam} = P'_1 \times T_R + P'_2 \times t_{MD} \quad (4.16)$$

$K_C=2$ olduğunda her bir literalın açılması için iki taraflı kesmede 6.5, bir taraflı kesmede (SKY) ise 7.5 kısa komut gerekir. Tablo 4.1'de görüldüğü gibi, sözü geden veri tabanında genellikle soldan kesme yöntemi(SKY) kullanıldığı için $t_R = 7.5 \times t_{ko} \times B$, burada t_{ko} bir kısa komutun sürdürülme süresi, $B = 1340$ bayt ise bir bloğun kapasitesidir. Bütün bunlar (4.16)'da dikkate alınırsa bloğa tam ulaşımın aşağıdaki sürede mümkün olacağı görülebilir.

$$T_{tam} = P'_1 \times T_R + P'_2 \times t_{MD} = 36.9\text{ms}$$

Yukarıda sıkıştırılmamış verilerin bir bloğuna ulaşım süresinin $T_{ge}=39.6ms$ olduğunu hatırlarsak, sıkıştırma sayesinde her bloğa ulaşım süresinde $39.6-36.9=2.7ms$ zaman tasarrufu elde edildiği görülebilir.

Genellikle, bilgi işlem sistemlerinde, merkezi işlemcinin her bir 10^6 işlemine karşın ortalama olarak 112×10^3 bayt giriş yapılmaktadır. Söz konusu sistemin gerçekleştirilmiş olduğu EC-1033 (IBM360) makinesinin hızı 0.2×10^6 işlem/san olduğu için girilen verilerin miktarı $0.2 \times 12 \times 10^3 = 22.4 \times 10^3$ bayt/san $=17$ blok/san. Böylece, bir saniyede, bir saat da ve bir nöbette tasarruf edilecek zaman miktarının 46×10^{-3} san, 165,6 san ve 0,367 saat olacağı beklenmektedir. EC-1033 makinesinin bir saat çalışmasının değeri 100rub olduğu için yıllık tasarrufun da yaklaşık olarak 10 000rub olacağı beklenmektedir.

İleriye dönük hesaplamalar göstermiştir ki, sanayi müesseselerinin pasaportları kontrol sistemi tam gücüne ulaştığında, elde edilecek olan tasarruf 80 000rub/yıl civarında olacaktır. Bundan başka, diskte tutulacak olan verilerin miktarı, sıkma neticesinde iki kat azalacağı için 57000 rub değerinde olan bir tane EC-5061 disk kurgusuna da tasarruf edilecektir.

4. Bölümün sonuçları

1. Önerilmiş olan sıkıştırma/açma yöntemlerinin (kortejleri soldan kesme yöntemi ve kortejleri iki taraftan kesme yöntemi) Assembler IBM360/370 dilinde tertiplenmiş olan programların esas nitelikleri belirlenmiş ve neyin yapılması ve nasıl yapılması gerektiği açıklanmış, bu programların kullanılması sonucunda bellek sisteminin değişik seviyeleri arasında veri miktarı dağılımının ve değişik seviyelere başvuru olasılıklarının nasıl değişeceği ve nasıl belirleneceği gösterilmiş, geliştirilmiş olan program paketinin sağladığı ve ileride sağlayabileceği ekonomik tasarrufun hesaplanma yolları gösterilmiştir.
2. Hesaplamalar göstermiştir ki, veri sıkıştırma paketinin uygulanması neticesinde Esas bellekle Maynetik disk belleği arasındaki veri akışı en az 2 kat azalmıştır ki bu da esas darboğaz olan sistem taşıtının işlemci tarafından daha çok kullanılabilmesini ve dolayısı ile sistemin performansını önemli bir ölçüde artmasını sağlamıştır.
3. Paketi uygulanması Maynetik disklerin sayısının azaltılmasına sebep olmuş, ele ilk aşamada değeri 57000 rub olan bir disk kurgusuna tasarruf edilmiştir.
4. Önerilmiş olan sıkıştırma/açma yöntemlerinden birinin donanımsal olarak gerçekleştirilmesi icat olarak Patent sertifikasına laik görülmüştür.
5. İçerisinde söz konusu program paketi olan OKS, Sovyetler birliği Alanlar arası komisyon tarafından, kendi alanında örnek sistem olarak bakanlar kurulu tarafından onaylanmaya sunulmuştur.

KAYNAKLAR

1. Sovyetler Birliđi kommunist partisinin XXIV kurultayının belgeleri. "Politizdat", 1981,s.223
2. EVREİNOV E.B. Homojen bilgisayar sistemleri, strüktürler, araçlar. Moskova, "Radyo ve İletişim",1981,s.208.
3. İVANOV A.P. Ekonomik problemlerin hesaplama parametreleri. Moskova, "İstatistik", 1976, s.168.
4. LOSLEBEN P. Veri yapıları, veri tabanları ve veri yönetimi. Bilgisayar sistemleri tasarımının otomasyonu, Veri tabanlarının modellenmesi. İngilizce'den çeviri, Moskova, "Mir", 1979,s.359-442.
5. MARTIN J. Bilgisayar sistemlerinde veri tabanlarının organizasyonu. İngilizce'den çeviri, Moskova, "Mir", 1980, s.664.
6. Genel Amaçlı Bilgi Sistemleri. Veri tabanlarını yönetim sistemlerinin analitik tahlili. İngilizce'den çeviri, Moskova, "İstatistik", 1975, s. 442.
7. DOLOTOV V.G. Otomatlaştırılmış kontrol sistemlerinin teknik ve bilgi desteđi. Moskova, MEİ, 1977, s.31.
8. KURBAKOV K.İ. Otomatlaştırılmış sözlükte bilgi kodlama ve arama. Moskova, " Sov. Radyo", 1968, s248.
9. KOROLEV M:A:, KLEŞKO G.N., MİŞENİN A.İ., Bilgi sistemleri ve veri yapıları. Moskova, "İstatistik", 1977, s. 183.
10. AKUŞSKİY İ.Y., ZABOLOTSKİY V.N. Bilgi sıkma ilkesine kombinatör yaklaşım. Bilgisayar ve Programlama. Seri 6, Moskova, "Sov. Radyo", 1971, s.5-17.
11. BOBKO İ.M. Otomatlaştırılmış kontrol sistemleri ve onların adaptasyonu. Novosibirsk, " Nauka", 1978, s.111.

12. RAMAMOORTHY C.V., WAH B.W. Data management in distributed data bases. "AFIPS Pres, Montvale, New Jersey, 1979, vol. 48, s.667-680.
13. SAMOYLOV L. K. Çağdaş veri toplama ve sıkıştırma sistemlerinin imkanlarının değerlendirilmesi. "Otomatlaştırma ve Bilgisayar", 1982, No 4, s.14-17.
14. İLİÇEV V.M. Otomatlaştırılmış kontrol sistemlerinde sıkışık veri saklama yöntemlerinin araştırılması ve geliştirilmesi. Doktora Tezi, Moskova, MEİ, 1980, s.20.
15. DVİNOGORTSEV G:P, YAŞİN V.M. Bilgisayar sistemlerinde bilgi alışverişi sistemleri ve donanımları. Moskova, "İletişim", 1976, s. 216.
16. Harrah D. A model for applying information and utility functions Philosophy of Sc.,1963, vol. 30, No 3, p. 267-273.
17. TSUKERMAN İ.İ. Beyine ve bilgisayara bilgi girişi hakkında. İnformasyon ve Siberetik, Moskova, "Sov. Radyo" 1967, s.199-215.
18. POSPELOV G.S. Bilgisayar kullanımına yeni yaklaşım. Siberetik, Sınırsız imkanlar ve olabilecek sınırlamalar, Gelişme Perspektifleri. Moskova, "Nauka", 1981, s.76-82.
19. ŞREYDER Y.A. Bilişim teorisinin semantik yanları. Enformasyon ve Siberetik, Moskova, "Sov. Radyo", 1967, s. 15-47.
20. ŞTARKOV Y.M. Bilişim teorisinde veri fazlalığının giderilmesi problemleri. Karmaşık sistemlerde kodlama, Moskova, Sovyetler birliği bilim akademisi, Siberetiğin problemleri üzere bilim şurası, 1974, s.154-163.
21. VENÇOVSKİY L:B. Teknik- Ekonomik bilgileri kodlama yöntemleri. Moskova, "Sov. Radyo",1978, s.120.
22. SOLOVEV V.V. Mesajın iletişim sırasında iyi kodlanması. Moskova, "Enerji", 1970, s.64.

23. ABİDOV Ç.C, KAHRAMANOV Ş.Ş. Otomatlaştırılmış bölgesel sistemin veri tabanının jenerasyonu. Azerbaycan üniversitelerinin doktora öğrencilerinin II ulusal konferansı, Baku, 1979, s.196.
24. KAHRAMANOV Ş.Ş. Veri sıkıştırma problemi. Azerbaycan üniversitelerinin doktora öğrencilerinin IV ulusal konferansı, Baku, 1981, s.20-23.
25. KAHRAMANOV Ş.Ş. Verilerin hızlı sıkıştırılması ve açılması. Azerbaycan üniversitelerinin doktora öğrencilerinin IV ulusal konferansı, Baku, 1981, s.131.
26. SAKEROV T.M, KAHRAMANOV Ş.Ş. EC EVM için ikili verileri sıkıştırma ve açma sistemi. Bilgi toplama ve işleme, Ç. Yıldırım adına AzPİ'nin bilimsel çalışmalar dergisi. Baku, 1981, s.95-98.
27. KAHRAMANOV Ş.Ş. Tablo verilerinin mantık cetvelleri yöntemi ile sıkıştırılmasının değerlendirilmesi, Ç. Yıldırım adına AzPİ'nin bilimsel çalışmalar dergisi. Baku, 1981, s.123-126.
28. KAHRAMANOV Ş.Ş. Sanayi müesseselerinin pasaportları veri tabanında veri sıkıştırma ve açma. Bilgi toplama ve işleme, Ç. Yıldırım adına AzPİ'nin bilimsel çalışmalar dergisi. Baku, 1981, s.100-104.
29. KAHRAMANOV Ş.Ş. Veri sıkıştırma ve açma yöntemleri kompleksinin seçilmesi. Azerbaycan üniversitelerinin doktora öğrencilerinin V ulusal konferansı, Baku, 1981, s.131.
30. KAHRAMANOV Ş.Ş. İlişkisel veri tabanlı RAMİS sisteminin tasarlanması. Devlet kayıt No 79047735, Arşiv No B998705, AzPİ, 1980, s. 79.
30. KAHRAMANOV Ş.Ş. İlişkisel veri tabanlı RAMİS sisteminin tasarlanması. Devlet kayıt No 79047735, Arşiv No B998705, AzPİ, 1980, s. 79.
31. KAHRAMANOV Ş.Ş. İlişkisel veri tabanlı RAMİS sisteminin tasarlanması. Devlet kayıt No 79047735, Arşiv No B905691, AzPİ, 1980, s. 187.

32. POZDNYAK G.E. Etkili bölgesel dağıtılmış bilgisayar sistemlerinin tasarım problemleri. Bilgisayar Sistemleri, Ser. 1. Moskova, "İstatistik", 1980, s. 25-40.
33. MAMZELEV İ. A, PŞENİÇNİKOV A. P. Dağıtılmış bilgi gönderme , işleme ve saklama sistemlerinin yapısal problemleri. Bilgisayar sistemleri, Ser. 1, Moskova, "İstatistik", 1980, s.50-57.
34. KANIGIN Y.M. İdare etmede kağıtsız teknoloji. Siberetik. Sınırsız imkanlar, olabilecek sınırlar. Moskova. "Nauka", 1981, s. 27-34.
35. EVREİNOV E.V. Tek bir dağıtılmış bilgisayar sistemi. Problemler ve gelişme perspektifleri. Bilgisayar sistemleri, Ser .1, Moskova, "İstatistik", 1980, s.8-25.
36. BELONOGOV G.G., KOTOV R.G. Otomatlaştırılmış bilgi arama sistemleri. Moskova, "Sov. Radio", 1968, s. 184.
37. GENDEL B.G., LEVİN N.A. Otomatlaştırılmış kontrol sistemlerinde bilgi işlem teknolojisinin optimizasyonu. Moskova, " İstatistik", 1977, s.232.
38. BRAVERMAN E.M., KİSELEVA N.E., MUÇNİK İ.B., NOVİKOV S.G. Büyük bilgi kütüklerinin işlenilmesine lingivistik yaklaşım. Otomasyon ve Telemekanik, 1974, No 11, s.73-88.
39. LİTVAKOV B. M. Birakılmış verileri olan büyük sayısal kütüklerin araştırılması. Otomasyon ve Telemekanik, 1974, No 11, s.73-88. Otomasyon ve Telemekanik, 1975, No 8, s.81-92.
40. AVEN P.O. KİSELEVA N.E., MUÇNİK İ.B. Veri matrislerinin kesişen bloklarla aproksimasyonu. Otomasyon ve Telemekanik, 1980, No 8, s.122-132.
41. SLETKOV S.N., VOLKOV B.G. Bilgi-Mantık sistemlerinde verilerin saklanma ve aranma organizasyonu. Moskova, " Sov. Radyo", 1971, s.224.
42. PENİN P.İ. Sayısal bilgileri iletme sistemi. Moskova, "Sov. Radyo", 1976, s.368.

43. ŞELİKOV A. A., SELİVANOV Z. P., Bilgisayarlar, El kitabı, Moskova, "Enerji", 1978, s.224.
44. KLUBA V.V., MAMİKONOV A.G., ŞELKOV A.B. Otomatlaştırılmış kontrol sistemlerinde program modüllerinin açılması. Otomasyon ve Telemeklik, 1980, NO 8, s. 133-141.
45. HOOR K. Kayıtların işlenmesi. Programlama dilleri, İ İngilizce'den tercüme, Moskova, "MİR", 1972, s. 278-343.
46. DAL U.İ. Kesintili olayları modelleme dilleri. Programlama dilleri, İngilizce'den tercüme, Moskova, "MİR", 1972, s. 344-403.
47. İktisadiyatta matematik ve sibernetik. El kitabı, Moskova, "Ekonomik", 1971, s. 224.
48. Bilgisayar, bilgi işleme. El kitabı, Bakanlar kurulu, 1970, s.248.
49. ZELKOVİTS M., ŞOU A., GENNON. Program sisteminin tasarım prensipleri, İngilizce'den tercüme, Moskova, "Mir", 1982, s. 368.
50. ŞOME J. Veri bankaları. Fransızca'dan tercüme, Moskova, " Energoizlat", 1981, s. 72.
51. FLORES A. Temel programlama desteği. İngilizce'den tercüme, Moskova, "Mir", 1971, s. 520
52. SVİRİDENKO V.A. Sıkıştırılmış veri bulunduran sistemlerin araştırılması. Moskova, "Svyaz", 1977, s. 184.
53. KNUT D. Bilgisayar için programlama sanatı. Cilt 3, İngilizce'den tercüme, Moskova, "Mir", 1978, s. 884.
54. TEMNİKOV F. E., AFONİN V. A., DİMİTRİYEV V.İ. Bilişim teknolojisinin teorik temelleri. Moskova, "Energia", 1971, s. 424

55. SHYBERT R.F. Basic concepts in data management systems, Datamation, July, 1972, p. 42-47.
56. YAGLOM A.M., YAGLOM İ.M. Olasılık ve bilişim. Moskova, "Nauka", 1973, s. 511.
57. DEVINSSON L. Veri sıkıştırma sistemlerinin teorik analizi. İngilizce'den tercüme, IEEE, 1968, cilt 56, No 2, s.46-58.
58. NOVIKOV G.İ. Adresleri segmentleme yolu ile mikrokomutları adresleme masraflarının minimumlaştırılması. "Sovyetler birliği yüksek öğretim kurulunun haberleri, Cihazcılık", 1974, No1, s. 71-73.
59. ASKEROV T.M. Bilgi arama sistemlerinde uygulanan mantık cetvellerini sıkıştırma ve açma algoritmaları. Bilgisayar ve Programlama, Ser.8. Moskova, "Sov. Radyo", 1974, s. 100-122.
60. TCIMBAL V.P. Kodlama teorisinde problem çözümü. Kiev, "Vişa Şkola", 1976, s.276.
61. BABKİN V.P., KRYUKOV A.V., ŞTARKOV Y. M. Veri sıkıştırma. Uzay Araştırma Cihazcılığı, Moskova, "Nauka", 1972, s.172-209.
62. VİTTİH V.A. Deneysel araştırmalarda veri sıkıştırma. Siberetik Problemler, Moskova, Sovyetler birliği bilim akademisi, 1974, s. 5-23.
63. COOPER W.S. The storage problem. Mech. Translat, 1958, No 2, p. 74-83.
64. DOLOTOV V.G., İLİÇEV B.M Otomatlaştırılmış kontrol sistemleri için veri sıkıştırma yöntemleri kompleksinin seçilmesi. MEI, Ser. 380, 1978, s.90-93.
65. DOLOTOV V.G., İLİÇEV B.M Otomatlaştırılmış kontrol sistemlerinde veri sıkıştırma için sınıflandırma diyagramı. Otomatlaştırılmış Kontrol Sistemlerinin Donanım ve Yazılım Araçları, Moskova, 1978, s.144-146.

66. TCVETKOV V.N. "Minsk" bilgisayarlar için matematik destek. Moskova, "İstatistik", 1974, s. 328.
67. IBM/360 bilgisayarı. Çalışma prensipleri. İngilizce'den tercüme, Moskova, "Sov. Radyo", 1969, s.440.
68. SADOVNİKOV V.İ., EPŞTEYN V.L. Kontrol sistemlerinde bilgi akışları. Moskova, "Enerji", 1974, s.240.
69. Williams R. On application of relational data structures in computer graphics- Informat. Processing, 1974, IFIP Gong., Stochosm, p.153-165.
70. CHANG S.K, DONATO N. Mc CORMICK V.H. A relational data base for pictures.- Proc.Workshop on data bases for interact. des., Univ. of Waterloo, Canada, 1975, p. 142-149.
71. BANDURSKI A., JEFFERSON D. Enhancements to the BBTG model for computer aided chip design- Proc. Workshop on data bases for Interact. des., Univ. of Waterloo, Canada, 1975, p. 215-230.
72. VUL V.A. Grafik etkileşim sistemleri ve uygulamaları. Harici Radyo Elektronik, 1980, No 1.
73. ABİDOV Ç.C. Sanayi müesseselerinin pasaportları için veri tabanının araştırılması ve geliştirilmesi. Doktora Tezi, AzPETROLKİM üniversitesi, 1892, s. 178.
74. KERSHBERY J., KLUG A., TSUHITGIS D. A. Taxonomy of data models. System for change data bases.-North Holland publishing company, 1976, p.43-64.
75. KOPEV V.V., ÇETVERKOV V.N. Yüksek ifadelilik niteliği olan veri modelleri. Lojik Kontrol, Ser. 2, "Energoizdat" 1981, s. 122-126.
76. KOPÇEV V.V., POSUDİN V.N. Bilişimsel Yapıların Karmaşıklığı. Büyük sistemlerin kelimesinin ve etkisinin ölçülmesi. Moskova, 1980, s.54-84.

77. ŞİĞİN A.G., DERYUGİN A.A. Bilgisayarlar (Bellek sistemi). Moskova, "Enerji", 1975, s. 536.
78. ASKEROV T. M., ABİDOV Ç.C. Bölgelerarası bilişim sisteminin yapısı. 2. Bilim-Teknik sempozyumu, Suzdal, 1981, s.43-45.
79. RUDZİTSKİY B.M. İlişkisel sistemler. Bilgisayar Sistemleri, Ser 1, Moskova, "İstatistik", 1980, s. 98-107.
80. GLUŞKOV V.M. Sayısal otomatların sentezi. Moskova, "Fizmatgiz" 1962, s. 476.
81. MİLLER R. Anahtar fonksiyonlar teorisi. Cilt 1, İngilizce'den tercüme. Moskova, "Mir", 1970, s. 416.
82. KOJURİN F.D., YARMOŞ N.A. Büyük bilgi kütüklerinin biçimsel işlenmesi. Minsk, 1973, s. 200.
83. ALFEROVA Z.V. Graf teorisi kullanmakla bilgisayarlar için ekonomik hesaplamalar yapan program desteği. Moskova, "İstatistik", 1974, s. 208.
84. GOLUNKOV Y.V. Mikroprogramların algoritmik tamlığı ve karmaşıklığı. Sibernetik, 1977, No 3, s. 1-15.
85. MAYOROV S.A., NOVİKOV G.İ. Bilgisayarların yapı prensipleri. Leningrad, "Maşinostroenie", 1974, s. 432.
86. BARANOV S.İ. Mikroprogramlı otomatların sentezi. Leningrad, "Enerji", 1974, s. 216.
87. TRAKTENGERTC E.A. Otomatlaştırılmış kontrol sistemlerinin program desteği. Moskova, "İstatistik", 1974, s.288.
88. LAZAREV V.G., PİYİL E.İ. Kontrol otomatlarının sentezi. Moskova, "Enerji", 1970, s. 448.

89. NECEFOV E.M., KAHRAMANOV Ş.Ş. Mikroprogramlı otomatların sentezi hakkında. AzDU, Bilimsel notlar No 1, Baku, 1974, s. 91-97.
90. DYAÇENKO V.V. Algoritmaların mantıki şemasını tertipleme yöntemi. Kontrol Kurguları ve Sistemlerinin Tasarımı. Moskova, 1974, s.10-16.
91. KUZNETCOV O.P., ADELSON-VELSDKİY G.M.. Mühendisler için matematik . Moskova, "Enerji", 1980, s.384.
92. KORŞUNUV Y.İ. Sibernetiğin matematik temelleri. Moskova, "Enerji", 1980. s.424.
93. NOVİKOV G.İ. Mikroişlem sisteminin ve mantık şartlarının hiyerarşisi. Sayısal Kurguların Optimizasyonu. Leningrad, 1976, s. 70-84.
94. POSPELOV D.A. Bilgisayar sistemine giriş. Moskova, "Sov. Radyo", 1972, s. 280.
95. RUMŞİNSKİY L. Z. Olasılık teorisinin temelleri. Moskova, "Nauka", 1976, s.240.
96. KENUY M. G. Çabuk istatistik hesaplamalar. El kitabı, Moskova, "İstatistik", 1979, s. 69.
97. MAKDUGAL M. Sistem seviyesinde modelleme. Bilgisayar Sistemi Tasarımının Otomasyonu, İngilizce'den çeviri, Moskova, "Mir", 1979, s. 11-121.
98. NAYOROV S. A., NOVİKOV G.İ. Bilgisayarların yapısı. Leningrad, "Maşinostroenie", 1979, s. 384.
99. ULANOV G. İ., ALİEV R. A., KRİVOŞEEV V.P. Sanayi müesseseleri için otomatlaştırılmış sistemlerinin geliştirilmesi yöntemler. Moskova, "Energoatomizdat", 1983, s. 320.

EK

“ONAYLIYORUM”

Azerbaycan Cumhuriyeti Devlet Planlama Teşkilatı
Başkanının Birinci Yardımcısı

(İmza ve Mühür)

F.D. RUSTAMBEYOVA

28 Kasım 1980

TUTANK

Bölgeler arası otomatlaştırılmış bilgi sisteminin sanayi uygulamasının alanlar arası
komisyon tarafından tespiti ve kabulü

28 Kasım 1980

Komisyon Üyeleri:

BAŞKAN

Memmedov F. N. Azerb. Cümhur. DPT'nin bilim ve teknik şöbesinin başkanı

ÜYELER:

- | | |
|-------------------|---|
| Aleskerov G.M | - Azerbaycan DPT'nin İktisat-Matematik Şöbesi'nin başkanı |
| Alhanov G.G | - Azerbaycan Bilişim Enstitüsü'nün müdür yardımcısı |
| Arakelov T.M | - Azerbaycan Bilişim Enstitüsü'nün söbe müdürü |
| Askerov T.M | - Azerbaycan Teknik Üniversitesi'nin Bilgisayarlar Bölümü başkanı |
| Baranovski Y.İ | - Sovyetler Birliği Devlet Standartları Enstitüsü'nün söbe müdürü |
| İvanov İ. V. | - Sovyetler Birliği Bilişim Enstitüsü'nün söbe müdürü |
| İbrahimov T.A. | - Azerbaycan Bilişim Enstitüsü'nün söbe müdürü |
| Kasimov T.G. | - Azerbaycan Bilişim Enstitüsü'nün müdür yardımcısı |
| Neçayev V.P. | - Sovyetler Birliği Bilişim Enstitüsü'nün söbe müdürü |
| Memetzade P.N. | - Azerb. Bilim Akademisinin Jeoloji Enstitüsü'nün söbe müdürü |
| Mahmudov Y.A. | - Azerb. Sibernetik Enstitüsü'nün müdür yardımcısı |
| Mehtiyev D.M | - Azerbaycan Bilişim Enstitüsü'nün müdürü |
| Nezameykov İ.İ. | - Sovyetler Birliği Devlet Bilim ve Teknik Konseyi'nin Baş İdaresinin bilim danışmanı |
| Savosin V.G | - Sovyetler Birliği Devlet Bilim ve Teknik Konseyi'nin söbe müdürü |
| Şatberaşvili O.B. | - Gürcistan Bilişim Enstitüsü'nün müdürü |

Komisyonu Sunulmuş olan belgeler:

1. Bölgelerarası otomatlaştırılmış bilgi sisteminin(BOBS) tasarımı ve uygulamasına ait teknik şartname.
2. BOBS'nin teknik – iş projesi
3. BOBS'nin tasarım ve uygulama yöntemleri
4. BOBS'nin kapsadığı problemler hakkında alanlar arası komisyonun tutanağı
5. Sovyetler Birliği Bilim Akademisinin Merkezi İktisat-Matematik Enstitüsü ile Azerbaycan DPT'nin Bilişim Enstitüsü arasında yapılmış olan anlaşma
6. Azerbaycan DPT'nin Bilişim Enstitüsü ile Azerbaycan Teknik Üniversitesi arasında yapılmış olan anlaşma
7. Azerbaycan Teknik Üniversitesi ile Azerbaycan DPT'nin İktisat Enstitüsü arasında yapılmış olan anlaşma
8. Sovyetler Birliği Bilim Akademisi Senatosunun 01.02.1977 tarihli, No 10103-240 kararı
9. Sovyetler Birliği Devlet Bilim ve Teknik Konseyi'nin 09.06.1978 tarihli No 255 kararı
10. 0.80.02 problemi hakkında Azerbaycan Bakanlar Kurulu'nun 19.11.1978 tarihli No 405 ve 20.11.1979 tarihli No 465 kararları

Komisyon, Proje yöneticisi Dr. Kasimov F.G'nin ve program geliştiricisi Abidov Ç.C'nin BOBS'in sanayi uygulamasına hazır olduğu hakkındaki raporlarını dinledi.

BOBS'un Nitelikleri

1. BOBS projesi teknik taleplere, standartlara ve belgelendirme taleplerine tam olarak uyularak bitirilmiş bulunmaktadır.
2. BOBS'ın temelinde sanayi müesseselerinin pasaportlarını ve teknik-iktisadi bilgileri bulundurmakta olan veri tabanı durmaktadır. Bu bilgilere ulaşım üç seviyede gerçekleştirilmiş durumdadır: İttifak (Sovyetler Birliği) seviyesi, Cumhuriyet seviyesi ve Alan seviyesi
3. BOBS'ın geliştirilme amaçları aşağıdakilerdir:
 - Çabuk verilebilen, güvenli olan ve son durumu yansıtan bilgiler vermekle yönetimin kalitesini yükseltmek işinde her seviyeden olan yöneticilere yardımcı olmak
 - Bilgisayar sistemlerinin ve ilgili araç gereçlerin verimliliğini yükseltmek ve bu yolla daha çok alanı kapsamak
4. BOBS, gelecekte genişletilebilme amacı ile karışık yapılı veri tabanını destekleyecek olan mantıki seviyede belge- faktografik arama sistemi ile donatılmıştır.
5. BOBS'un veri tabanı aşağıdaki prensipler esasında kurulmuştur:
 - Verilerin bir kere girilip çok defa ve çok amaçlı kullanılabilmesi
 - Karmaşık talepleri basit taleplerin kompozisyonu olarak işlemekle pratik olarak istenilen büyüklükteki taleplere cevap verebilme
 - Verilerin fazlalıklardan arındırılması
 - Verileri müsaide edilmeyen müdahalelerden koruma
 - Verilerin problemlerden bakımsız olarak merkezleştirilmiş entegrasyonu
 - Başka bölgelerin veri tabanı ile entegre olabilme

▪ Kullanıcı ile etkileşim

6. BOBS'ın ilk sürümünün esas görevleri aşağıdakilerdir:
 - Plan organlarına müesseselerin stok-iktisadi ve dinamik durumları ile ilgili bilgiler vermek
 - Kurumlar ve müesseseler arasında talebe göre bilgi akışını sağlamak
 - Problemlerin tespiti ve çözümü için yönetici personele gereken bilgiler
 - Diğer bölgelerle tam entegrasyon ortamında çalışabilmek
7. BOBS, üç rejimde çalışabilir: Bilginin seçilmiş olan hedefe ulaştırılması, Bilginin çoklu hedefe ulaştırılması, Diyalog
8. Kullanıcılara verilen yanıtlar tipik belgeler ve tablolar şeklindedir
9. BOBS, çağdaş bilim- teknik seviyeye uygun olarak geliştirilmiştir
10. BOBS için tertiplenmiş olan teknik belgeler sanayi uygulaması için yeterli seviyede bulunmaktadır.
11. BOBS'un matematik-lingivistik desteği çağdaş taleplere cevap vermekte ve değişik tip veri tabanlarını destekleyebilir niteliktedir
12. BOBS'un teknik alt yapısı onun görevlerinin tam olarak yerine getirebilmesi için yeterli değildir.

KOMİSYONUN KARARI

Bölgelerarası otomatlaştırılmış bilgi sistemi (BOBS) sanayi uygulaması için kabul edilsin

KOMİSYONUN ÖNERİLERİ

1. BOBS'u Sovyetler Birliği Baş Bilim ve Teknik Konseyi'ne örnek proje olarak nitelendirilmesi önerisinde bulunulsun
2. BOBS'un veri tabanı alanlar arası problemlerin çözümünde aktif olarak kullanılsın
3. Kurumların da BOBS'tan yararlanabilmesi için bakanlıklar ve kurum yöneticileri arasına gereken ilişkiler kurulsun
4. BOBS üzere baş idareye, bu sistemin il, ilçe ve bilgi-işlem merkezleri tarafından kullanılabilmesi için gereken metodik belgeleri hazırlama ricasında bulunulsun.

Komisyon Başkanı

(İmza)

Üyeler

(Toplam 15 imza)

ONAYLIYORUM

Azerb. Petrol-Kimya Sanayisi Bakanlığı
Bilgi İşlem Merkezinin Müdürü
F.İ. ŞERİFOV

19 Aralık 1981

BOBS (Bölgesel Otomatlaştırılmış Bilgi Sistemi) sisteminin kullanıcıya teslim edilmesi ve kullanmaya başlaması hakkında

TUTANAK

Biz, aşağıda imza atanlar: Azerb. Petrol-Kimya Sanayisi Bakanlığı Bilgi İşlem Merkezinin; Müdür Yardımcısı ZEYNALOV R.Y, Baş Mühendisi AHMETOV B.A., Otomasyon ve Programlama Söbesinin Müdürü KULİEV T.M. ve Tasarımcılar: Azerb. Teknik Üniversitesinin Bilgisayar Bölüm başkanı Prof. Dr. ASKEROV T.M., Baş Öğretmen ABİDOV Ç.C, Baş Öğretmen KAHRAMANOV Ş.Ş ve Azerb. Bilişim Enstitüsünün bilim üzere Müdür Yardımcısı KASIMOV F.G. bu tutanağı ondan dolayı tertitledik ki, Azerb. Teknik Üniversitesi ve Azerb. Bilişim Enstitüsü Bölgesel Otomatlaştırılmış Bilgi Sistemini teslim etti ve Azerb. Petrol-Kimya Sanayisi Bakanlığı Bilgi İşlem Merkezi bu sistemi sanayi uygulaması amacı ile kabul etti.

Sistemin uygulanmasından elde edilecek ekonomik kazancın bir yılda 11825 ruble olacağı beklenmektedir.

Sistemi kabul edenler

(2 imza)

Sistemi teslim edenler

(6 imza)

ONAYLIYORUM

SOYUZNEFTEMAŞ

Bilgi İşlem Merkezinin Müdürü

F.C. NURİYEV

1 Kasım 1981

BOBS (Bölgesel Otomatlaştırılmış Bilgi Sistemi) sisteminin kullanıcıya teslim edilmesi ve kullanmaya başlaması hakkında.

TUTANAK

Biz, aşağıda imza atanlar: Siparişçi SOYUZNEFTEMAŞ 'ın temsilcileri: Bilgi İşlem Merkezinin Müdür Yardımcısı BAĞIROV C.A., Söbe Müdürü MUSAYEV F.A. ve Tasarımcılar: Azerb. Bilişim Enstitüsünün Müdür Yardımcısı KASIMOV F.G., Söbe Müdürü MELIKOV Y.A., Götürücü mühendis LIFANTEVA D.M, Azerb. Teknik Üniversitesinin Bilgisayar Bölüm başkanı Prof. Dr. ASKEROV T.M., Baş Öğretmen ABİDOV Ç.C, Baş Öğretmen KAHRAMANOV Ş.Ş bu tutanağı ondan dolayı tertipledik ki, Azerb. Teknik Üniversitesi ve Azerb. Bilişim Enstitüsü Bölgesel Otomatlaştırılmış Bilgi Sistemini teslim etti ve SOYUZNEFTEMAŞ 'ın Bilgi İşlem Merkezi bu sistemi tecrübe uygulaması için ile kabul etti.

Sistemin uygulanmasından elde edilecek ekonomik kazancın bir yılda 7.2 bin ruble olacağı tahmin edilmektedir

Sistemi kabul edenler

(2 imza)

Sistemi teslim edenler

(4 imza)